Date of acceptance Grade

Instructor

Unsupervised Learning for Image Classification

Yao Lu

Helsinki December 8, 2015

UNIVERSITY OF HELSINKI
Department of Computer Science

HELSINGIN YLIOPISTO — HELSINGFORS UNIVERSITET — UNIVERSITY OF HELSINKI

Tiedekunta — Fakultet — Faculty	Laitos — Institution — Department
Faculty of Science	Department of Computer Science

Tekijä — Författare — Author
Yao Lu

Työn nimi — Arbetets titel — Title
Unsupervised Learning for Image Classification

Oppiaine — Läroämne — Subject
Computer Science

Työn laji — Arbetets art — Level	Aika — Datum — Month and year	Sivumäärä — Sidoantal — Number of pages
	December 8, 2015	52 pages + 0 appendices

Tiivistelmä — Referat — Abstract

This thesis is an investigation of unsupervised learning for image classification. The state-of-the-art image classification method is Convolutional Neural Network (CNN), which is a purely supervised learning method. We argue that despite of the triumph of supervised learning, unsupervised learning is still important and compatible with supervised learning. For example, in the situation where some classes have no training data at all, so called zero-shot learning task, unsupervised learning can leverage supervised learning to classify the images of unseen classes. We proposed a new zero-shot learning method based on CNN and several unsupervised learning algorithms. Our method achieves the state-of-the-art results on the largest public available labelled image dataset, ImageNet fall2011.

ACM Computing Classification System (CCS):
A.1 [Introductory and Survey],
I.7.m [Document and text processing]

Avainsanat — Nyckelord — Keywords
Unsupervised Learning, Convolutional Neural Networks, Deep Learning, Image Classification

Säilytyspaikka — Förvaringsställe — Where deposited

Muita tietoja — övriga uppgifter — Additional information

Contents

1 Introduction

Biological vision systems are remarkably powerful. We can recognize an old friend among a hundred of strangers, despite of the changes of pose, clothing, background and viewpoint. How is this possible? How does vision work?

Philosophers and scientists have been investigating and speculating on vision since hundreds of years ago. In his book *Principles of Philosophy* published in 1644, Descartes developed his over-simplified mechanistic theory of vision as illustrated in Figure 1. In 1966, researchers in MIT tried to solve vision in a summer and only found vision is much more difficult than what they once thought [Pap66]. Even to date, we still do not have a full understanding of biological vision.

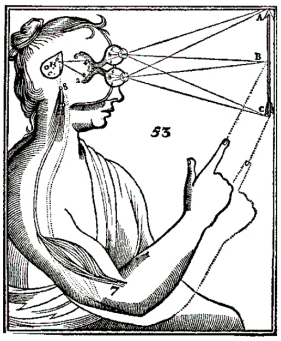


Figure 1: Descartes' theory of vision.[2]

Our knowledge of vision would enable us to build artificial vision systems [Fuk88, LBBH98]. Such vision systems can be employed for autonomous driving, image search, biometric recognition, surveillance and robots. On the other hand, in the process of building artificial vision systems, we could discover principles of vision, which in turn help us understand biological vision systems [CHY+14]. Building artificial vision systems has a long history of over 50 years [AT13], from early systems based on volumetric parts to modern ones based on artificial neural networks.

The primary goal of a vision system, either biological or artificial, is visual recognition, the ability to perceive an scene or object's physical properties such as shape, color and texture. According to [Per09], there are five visual recognition tasks, in order of difficulty:

- Verification. Test if an area of an image contains a particular object.
- Detection and localization. Finding where is a particular object in an image.
- Classification. Classifying an image into a number of categories.
- Naming. Name and locate all objects in an image.
- Description. Given an image, describe it by text.

[2]Image from *Principles of Philosophy* 1664.

In this thesis, we focus on the (image) classification task, which has been and still is an active research area in computer vision.

1.1 Problem Statement

The current state-of-the-art image classification method is Convolutional Neural Network (CNN) [LBBH98]. It has been employed as the workhorse for a number of image classification competitions [CMS12, KSH12, SLJ+15]. Moreover, CNN has already been employed widely in the industry, for example, Google and Baidu's content-based image retrieval system. It has been shown that CNN can even surpass humans in recognizing 1000 objects [SLJ+15].

However, there is a severe problem with CNN: it requires a large amount of labelled data. The acquisition of labelled data is very time-consuming and expensive. It requires human annotators to manually observe each data point and input the correct labels. For more specialized domain such as medical and satellite imaging, it requires experts to annotate the images. The number of qualified domain experts are rather limited, making the labelling even more expensive. And since the possible values that data can take are exponentially many, it is impractical to obtain a sufficiently large amount of labelled data.

In order to reduce the dependence of CNN on labelled data, we have to explore the realm of unlabelled data. Compared to labelled data, unlabelled data is numerous and can be obtained cheaply. Learning from unlabelled data is unsupervised learning. The pursuit of unsupervised learning for image classification started in 2006 [HOT06]. Despite of intensive research on this topic, current state-of-the-art image classification method, CNN, is a purely supervised learning method. Unsupervised learning methods such as sparse coding and pre-training are unnecessary for obtaining high-performance image classification [CMS12, KSH12, SLJ+15].

Is unsupervised learning a dead end for image classification? In a recent Nature paper [LBH15], neural network researchers LeCun, Bengio and Hinton pointed out this embarrassing situation. However, they still hold optimistic attitude towards unsupervised learning.

> *Unsupervised learning had a catalytic effect in reviving interest in deep learning, but has since been overshadowed by the successes of purely supervised learning...We expect unsupervised learning to become far more*

> *important in the longer term. Human and animal learning is largely unsupervised: we discover the structure of the world by observing it, not by being told the name of every object.* - LeCun, Bengio and Hinton

The integration of unsupervised learning for image classification remains elusive. Therefore, the central question in this thesis is:

> **While the state-of-the-art image classification methods are purely supervised, can unsupervised learning algorithms still be useful?**

1.2 Contributions

We showcase that unsupervised learning is useful in the context of zero-shot learning. Zero-shot learning is a (image) classification task in which some classes have no training data at all. The main contribution of this thesis is: based on the several unsupervised learning algorithms, we proposed a new zero-shot learning method, which is compatible to supervised learning. The effectiveness of our proposed method is demonstrated on the largest public available labelled image dataset, ImageNet 2011fall, which has over 14.2 million images and over 20000 object classes.

In Section 2, we review all the background knowledge which is necessary for understanding our zero-shot learning method. In Section 3, we formally introduce our zero-shot learning method. In Section 4, we conclude this thesis and discuss its implications and future research directions.

2 Background Knowledge

2.1 Image Classification

Image classification is a fundamental problem in computer vision. It is the task of classifying images into a number of categories. Image classification is the basis for higher level tasks such as object detection and image description, as explained in Section 1. It has been an active research area in computer vision over 50 years [AT13]. Despite of the efforts of 50 years, we still do not have a computer vision system which can achieve human level recognition ability for images in the wild. Image classification is difficult due to the variations inherent to images such as

translation, scaling, rotation and luminance changes. We divide image classification methods into two families: traditional methods and Convolutional Neural Networks.

2.1.1 Traditional Methods

The traditional image classification methods typically have four stages: feature extraction, quantization, pooling, and classifier.

Figure 2: Traditional image classification pipeline.

Feature Extraction Feature extraction is the first stage of the traditional image classification pipeline. Its job is to extract low-level discriminative local visual features such as edge and corner, which will be used for the next stage. Feature extraction is typically done in a convolutional or sliding window manner across the whole image (dense sampling). Different features are defined by different convolution kernels. See Figure 3 for an illustration of convolution.

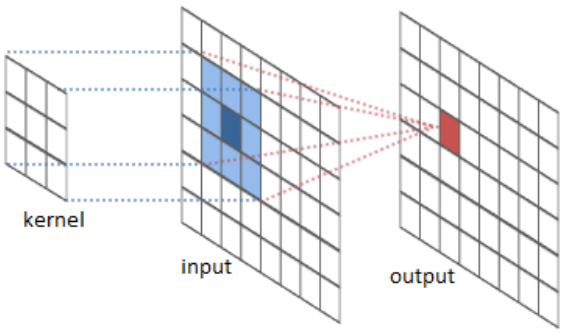


Figure 3: Convolution.[3]

A classical type of features is Gabor. A Gabor feature can be understood as a Gaussian envelope of a two-dimensional sinusoidal function,

$$g(x, y, \lambda, \theta, \psi, \sigma, \gamma) = \exp(-\frac{x'^2 + \gamma^2 y'^2}{2\sigma^2}) \cos(2\pi \frac{x'}{\lambda} + \psi) \quad (1)$$

$$x' = x \sin\theta + y \cos\theta, \quad (2)$$

$$y' = -x \cos\theta + y \sin\theta, \quad (3)$$

[3]Image from `http://colah.github.io/posts/2014-07-Understanding-Convolutions/`

where x and y are image position, λ is the wavelength parameter of the sinusoidal, θ is the orientation parameter, ψ is the phase parameter, σ is the standard deviation parameter of the Gaussian envelope and γ is the ellipticity parameter of the Gabor function. From Figure 4, we can see Gabor features are sensitive to direction and frequency.

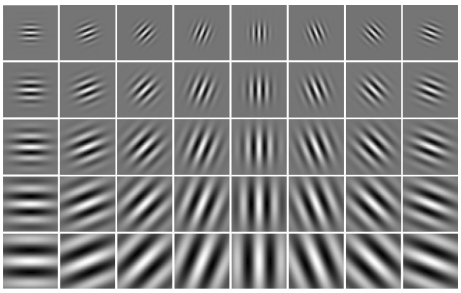


Figure 4: Gabor features.[4]

Another well-known type of features is scale-invariant feature transform (SIFT) [Low99]. SIFT features have robustness to scale, rotation and illumination changes. They based on histograms of image gradients. Image gradient is a directional change in the intensity or color in an image. It is used for extracting information such as edges. To obtain the image gradient along horizontal (x-axis) and vertical directions (y-axis), apply the following filters to an image, respectively.

$$\mathbf{G}_x = [-1\ 0\ 1] \quad \mathbf{G}_y = [-1\ 0\ 1]^T \tag{4}$$

The results of the filtering can be seen in Figure 5. The gradient magnitude is

$$\mathbf{G} = \sqrt{\mathbf{G}_x^2 + \mathbf{G}_y^2} \tag{5}$$

and the gradient orientation or direction is

$$\mathbf{\Theta} = \text{atan2}(\mathbf{G}_y, \mathbf{G}_x) \tag{6}$$

The feature extraction process of SIFT works as follows: For an image position, a 16x16 neighbourhood is taken. It is divided into 16 sub-blocks of size 4x4. For each sub-block, 8 bin histogram of image gradient orientations is created. As a result, a total of 128-dimensional feature vector is created. See Figure 6 for illustration.

A more recent approach is unsupervised feature learning including sparse coding [OF97] and Independent Component Analysis (ICA) [HKO04]. For example, the

[4]Image from `http://www.mathworks.com/matlabcentral/fileexchange/44630-gabor-feature-extraction`

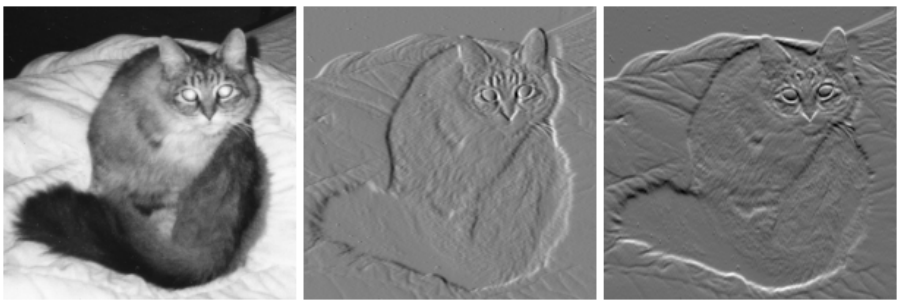


Figure 5: Image gradient. Left: original image. Middle: filtered image by applying $\mathbf{G}_x$. Right: filtered image by applying $\mathbf{G}_y$.[6]

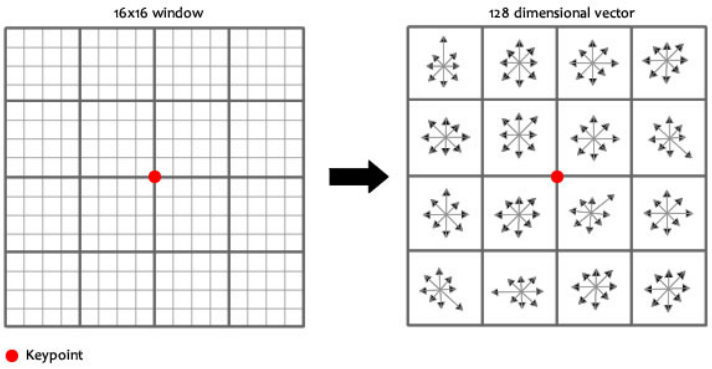


Figure 6: SIFT feature.[7]

features learned by ICA on natural image patches resemble the Gabor features due to their orientation, scaling and frequency selectivity. The details of these algorithms will be discussed in the Section 2.2. For a comprehensive review of unsupervised feature learning, see [BCV13].

Quantization Quantization or codebook learning is to build high-level features by aggregating low-level features in the previous stage. Since the amount of low-level of features are typically enormous, a quantization method is needed so as to keep the number of features manageable.

The simplest way of doing quantization is using k-means clustering. Each low-level feature is then assigned to its nearest cluster centroid. Then the assignments of low-level features instead of their real valued vectors are passed to the next stage. A more sophisticated codebook learning method is Fisher vectors [PD07, PSM10].

[6]Image from `https://en.wikipedia.org/wiki/Image_gradient`.
[7]Image from `http://www.aishack.in/`.

Fisher vectors are based on Gaussian mixture model, which is a generalization of k-means clustering. Therefore, Fisher vectors can be seen as an extension of the k-means clustering quantization.

Pooling Pooling is to aggregate the local features for whole image representation by a single vector. The simplest way of pooling is bag-of-features [FFP05], which uses the histogram of the high-level features as the vector representation of an image, as in Figure 7. However, bag-of-features pooling ignores the spatial relationship layout of the local features, which would cause ambiguity and loss in classification accuracy. An improved pooling method is spatial pyramid matching [LSP06], which uses bag-of-features at multi-scale grids of the image. The feature vectors created at multi-scale are then concatenated as a single vector. See Figure 8 for illustration.

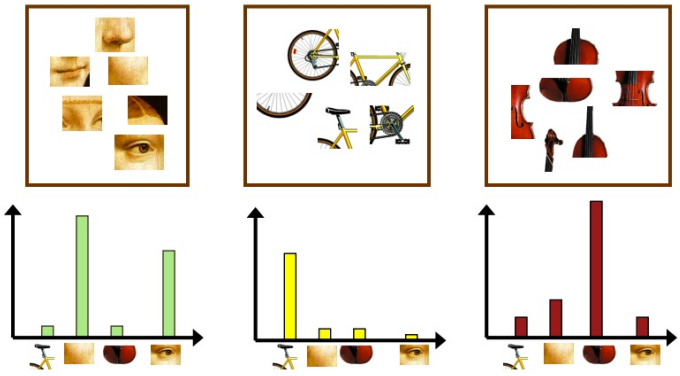


Figure 7: Bag of features.[8]

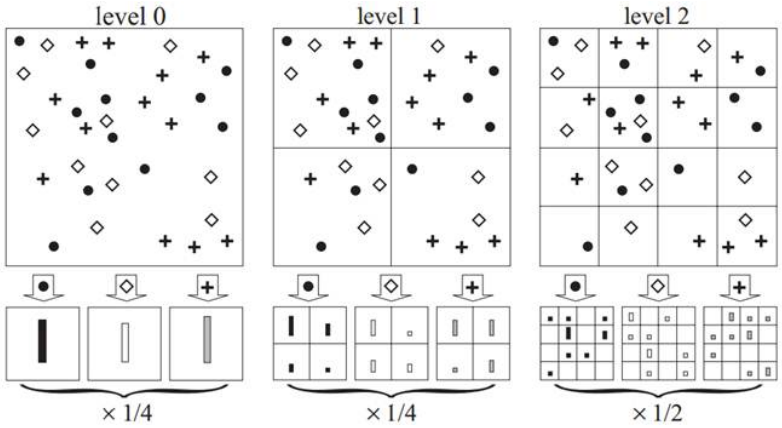


Figure 8: Spatial pyramid matching.[9]

[8]Image from `http://cs.brown.edu/courses/cs143/2011/results/proj3/senewman/`.
[9]Image from [LSP06].

Classifier The most popular classifier in the traditional image classification pipeline is support vector machine (SVM) [CV95], which is a linear classifier such that the decision function is

$$g(\mathbf{x}) = \text{sign}(f(\mathbf{x}))$$

where

$$f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b$$

The objective function of SVM (primal form) has the following

$$\min_{\mathbf{w}} \frac{1}{2}\|\mathbf{w}\| + C\sum_i l(\mathbf{x}_i, y_i) \tag{7}$$

where C is a hyper-parameter and $l(\cdot)$ is so-called hinge-loss and takes the form of

$$l(\mathbf{x}, y) = \max(0, 1 - yf(\mathbf{x}))) \tag{8}$$

Since $l(\cdot)$ is not differentiable, sub-gradient methods such as Pegasos [SSSSC11] are used for efficient optimization.

Kernel SVM is often used when the amount of data is not large. Kernel method makes SVM a nonlinear classifier, which could lead to higher classification accuracy. Typically used kernels include

- Histogram intersection kernel

$$K(\mathbf{x}, \mathbf{x}') = \sum_{i=1}^{N} \min(x_i, x_i') \tag{9}$$

- Gaussian kernel

$$K(\mathbf{x}, \mathbf{x}') = \exp(-\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{\sigma^2}) \tag{10}$$

- χ^2 (Chi-square) kernel

$$K(\mathbf{x}, \mathbf{x}') = \sum_{i=1}^{N} \frac{(x_i - x_i')^2}{x_i + x_i'} \tag{11}$$

It is difficult to decide beforehand which kernel to use in practice. The performance of each kernel depend on the data at hand. It is suggested to try all of them based

on validation error. Kernel SVM has the following objective function (dual form)

$$\min_{\boldsymbol{\alpha}} \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j K(\mathbf{x}_i, \mathbf{x}_j) - \sum_i y_i \alpha_i \tag{12}$$

$$\text{s.t.} \sum_i \alpha_i = 1, 0 \leq y_i \alpha_i \leq C \tag{13}$$

The decision function of kernel SVM is

$$g(\mathbf{x}) = \text{sign}(\sum_i \alpha_i K(\mathbf{x}_i, \mathbf{x}) + b) \tag{14}$$

When we have a large amount of high dimensional data, linear SVM would suffice for achieving high classification accuracy. For multi-class classification, one-vesus-rest strategy is typically used.

Summary In machine learning, it is a rule-of-thumb that features matter more than classifiers to classification accuracy [Dom12]. Good features need to be tolerant to transformations such as translation, orientation and viewpoint changes. On the other hand, good features also need to be discriminative so that they can distinguish different objects. One lesson that traditional image classification methods have taught us is the importance of local features. Global features of an image could change substantially even for slight transformations. For example, consider a global feature (filter) whose size is as large as the image. Even for only slight translations of the image, the output of the filter would change significantly. On the other hand, the response of a local feature would stay the same. Only the position of the response changes. Another lesson is: one can aggregate local features to form invariant high-level abstract features, which is critical to classification accuracy.

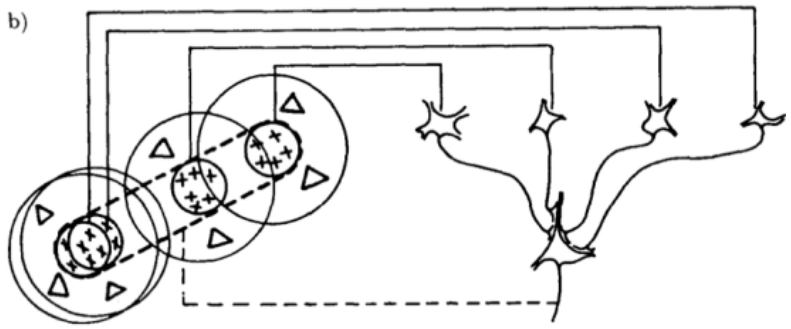
Traditional image classification methods were employed for winning several large scale image classification competitions such as ImageNet ILSVRC 2010 [LLZ+11] and ImageNet ILSVRC 2011 [SP11] competitions. they were the state-of-the-art image classification methods until CNN came into stage [KSH12].

2.1.2 Convolutional Neural Network (CNN)

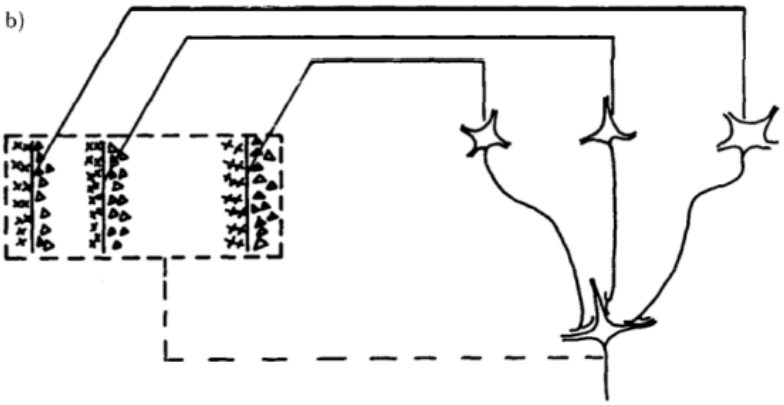
CNN shares a similar image classification pipeline as the traditional methods. However, while in the traditional image classification pipeline only the classifier is trained with labelled data, CNN is trained end-to-end, which means all the stages (feature

extraction, quantization, pooling and classifier) are learnable. CNN is the current state-of-the-art image classification method in various datasets such as MNIST [LBBH98], CIFAR-10 [KH09] and ImageNet [DDS+09].

The architecture of Convolutional Neural Network (CNN) rooted from the seminal work of the discovery of simple cells and complex cells in the brain by neuroscientists Hubel and Wiesel. Simple cells receive inputs from the receptive fields of LGN neurons. And complex cells the sum or pool the outputs from simple cells. See Figure 9 for illustration.



(a) Simple cell



(b) Complex cell

Figure 9: Hubel and Wiesel's cells.[10]

Inspired by Hubel and Wiesel, Fukushima proposed Neocognitron [Fuk88]. Neocognitron is a neural network model which uses artificial simple-cell layers and complex-cell layers alternatively, so as to achieve small translation invariance. However, Fukushima only proposed the Neocognitron architecture, without a learning algorithm. Therefore, Neocognitron is not a practical image classification method.

[10]Image from [Hub82].

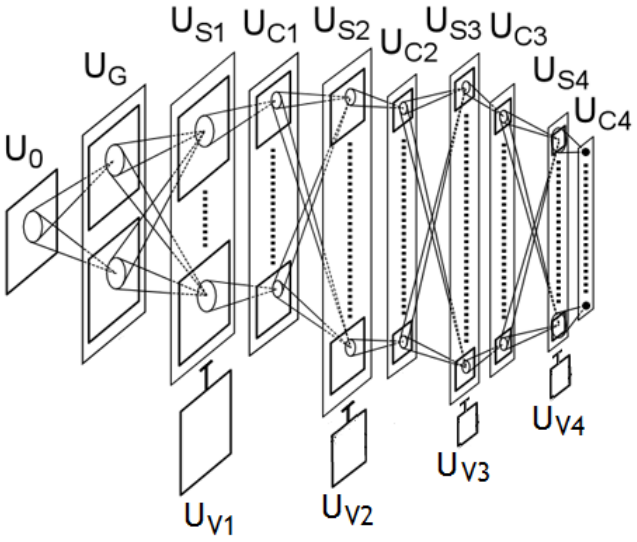


Figure 10: Neocognitron.[11]

LeCun [LBBH98] proposed CNN, which has a more simplified architecture as Neocognitron and a learning algorithm. LeCun applied the back-propagation algorithm [Wer82, RHW88] to learn all the weights of CNN and achieved the state-of-the-art results on MNIST. Moreover, he invented various tricks for speeding up the training and reduce the generalization error of CNN [LBOM12].

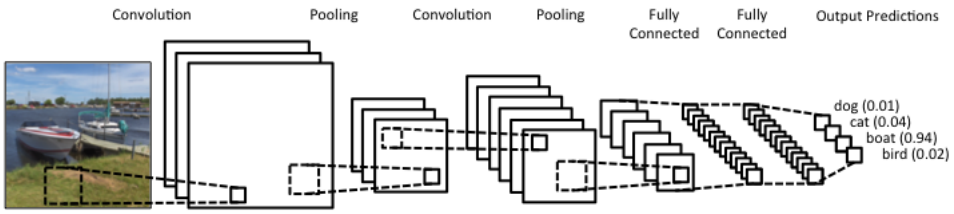


Figure 11: Convolutional Neural Network.[12]

Architecture In the following we list the key components of CNN architecture, including the classic ones and more recently ones

- Convolution layer. Convolution corresponds to the local feature extraction in the traditional image classification pipeline. The definition of two-dimensional

[11]Image from [Fuk88].
[12]Image from `www.clarifai.com`.

convolution is

$$y[i,j] = \sum_{m=0}^{M} \sum_{n=0}^{N} x[m,n]k[i-m, j-n] \tag{15}$$

where, for image position (x, y), $x[i, j]$ is the pixel value of the original image, $k[i, j]$ is the pixel value of the convolution kernel, $y[i, j]$ is the pixel value of the filtered image, M is the height of the original image and N is the width of the original image. Convolution is a heavy operation due to the size of images and the number of convolution kernels in CNN. Therefore, efficient convolution techniques are needed. There are two much techniques which are commonly used:

1. Conversion to matrix multiplication. Convolution can be formulated into matrix multiplication. Since there are many highly optimized matrix multiplication libraries such as OpenBLAS[13] for CPU and CuBLAS[14] for GPU, convolution can be done with higher speed than a naive implementation. To see how the conversion is possible, we give an example. For two matrices to convolute, the following equation holds

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \otimes \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & a_{21} & a_{22} \\ a_{12} & a_{13} & a_{22} & a_{23} \\ a_{21} & a_{22} & a_{31} & a_{32} \\ a_{22} & a_{23} & a_{32} & a_{33} \end{pmatrix} \begin{pmatrix} b_{11} \\ b_{12} \\ b_{21} \\ b_{22} \end{pmatrix}$$

   where $\otimes$ is the convolution operator.

2. Fast Fourier Transform (FFT). The convolution theorem allow us to convert a convolution operation into a multiplication operation with Fourier transform. For a matrix of of elements n, a naive Fourier transform algorithm has time complexity $O(n^2)$ but FFT only has time complexity $O(n \log n)$. For two matrices of elements m and n, the time complexity of naive convolution is $O(mn)$. And the time complexity of the FFT-based convolution is $O(m \log m + n \log n)$. If m and n are both large, we have a performance gain using the FFT-based convolution. However, in CNN, usually the convolution kernel are rather small. For example, 7×7, 5×5, 3×3 and even 1×1. However, if there are a large number convolution kernels, performance gain can be achieved, as shown in [MHL13].

[13] `http://www.openblas.net/`

[14] `https://developer.nvidia.com/cublas`

- Pooling layer. Pooling in CNN corresponds to the pooling in the traditional image classification pipeline. See Figure 12 for illustration.

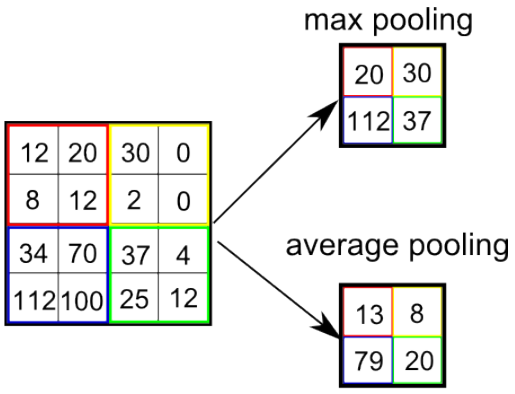


Figure 12: Max pooling and average pooling.[15]

There two commonly used pooling methods: average pooling and max pooling, as illustrated in Figure 12. Average pooling was proposed in original Le-Net architecture of CNN [LBBH98]. Max pooling was first used in the HMAX model [RP99], which is also a deep neural network with convolution. However, HMAX is not trained end-to-end as CNN. In [CMS12], CNN with max pooling was first used to win several visual recognition competitions [CMS12]. A theoretical analysis of the two pooling methods can be found in [BPL10].

- Fully connected layer. Fully connected layer is the same as the hidden layer in Multi-layer Perceptrons [RHW88]. It acts as a nonlinear transformation for the output classifier.

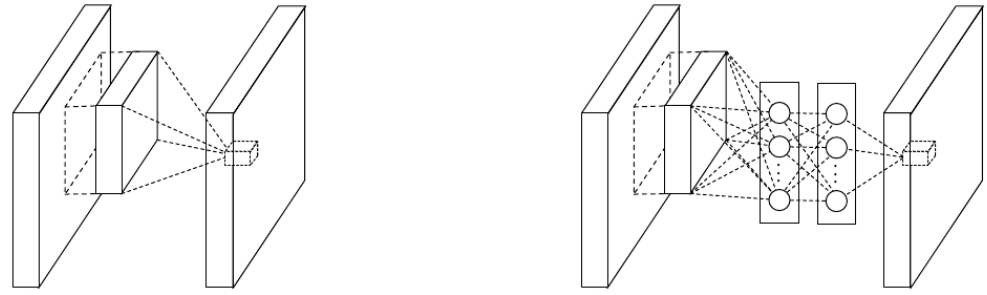
- Softmax layer. The softmax layer is the classifier of CNN. It takes inputs $\mathbf{y} = (y_1, ..., y_n)$ and outputs vector $\mathbf{x} = (x_1, ..., x_n)$ such that

$$x_i = \frac{\exp(y_i/T)}{\sum_j \exp(y_j/T)} \tag{16}$$

where T is the temperature parameter.

- Network in network layer. The convolution layer provides linear convolution. In order to achieve nonlinear convolution, we can replace the convolutional kernel, a matrix, with a Multi-layer Perceptron. This results in network in network [MLY14] layer, as illustrated in Figure 13.

[15]Image from `http://vaaaaaanquish.hatenablog.com/entry/2015/01/26/060622`.



(a) Linear convolution layer (b) Mlpconv layer

Figure 13: Network in network layer.[16]

- Inception. The inception layer is essentially a multi-resolution convolution layer. As illustrated in Figure 14, It concatenates 1×1, 3×3, and 5×5 convolution operations. It was first employed in GoogLeNet [SLJ+15] for winning the ImageNet ILSVRC 2014 competition.

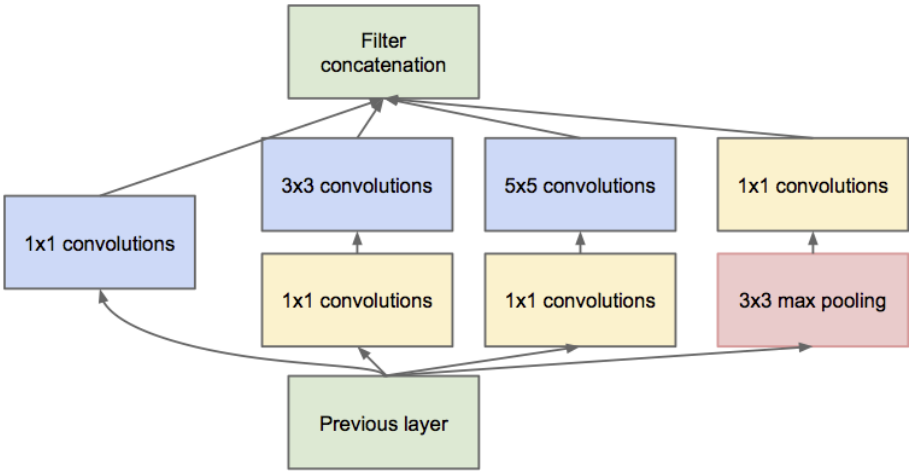


Figure 14: Inception layer.[17]

- Activation functions. There are several activation functions commonly used:

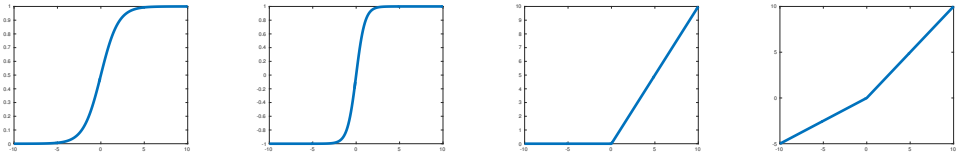
  1. Logisitic

$$f(x) = \frac{1}{1 + \exp(-x)}$$

  2. Hyper-tangent

$$f(x) = \tanh(x)$$

[16]Image from [MLY14].
[17]Image from [SLJ+15].



(a) Logistic (b) Hyper-tangent (c) ReLU (d) Leaky ReLU

Figure 15: Activation functions.

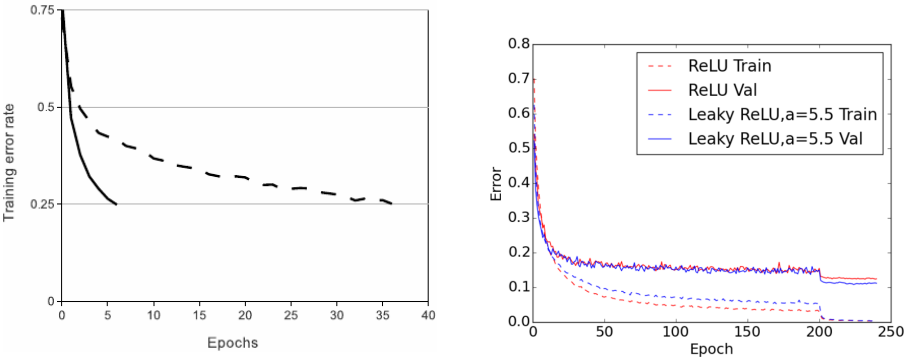
3. Rectifier linear unit (ReLU)

$$f(x) = \max(0, x)$$

4. Leaky rectifier linear unit (Leaky ReLU)

$$f(x) = \begin{cases} x, & x \geq 0 \\ \frac{x}{a}, & < 0 \end{cases}$$

where a is a parameter to be learned or fixed.



(a) ReLU (solid line) vs. Hyper-tangent (dashed line)

(b) ReLu vs. Leaky ReLU

Figure 16: Convergence of training a CNN on CIFAR-10 with different activation functions.[19]

It has been shown that ReLU gives much faster convergence than sigmoid functions such as logistic and hyper-tangent in training a CNN [KSH12], as in Figure 16 (a). And in [XWCL15], it has been shown that Leaky ReLU outperforms ReLU, as shown in Figure 16 (b).

[19]Image from [KSH12] and [XWCL15].

Training The training of CNN typically takes a large amount of time and computational resources compared to traditional methods. The success of training CNN depends highly on the hyper-parameters, which take great efforts to tune. Here we list several techniques which are commonly used.

- Back-propagation [RHW88]. This is the workhorse for training CNN. Back-propagation is a recursive method for calculating the gradient of the the cost function of a CNN.

- Stochastic gradient descent [RM51]. Instead of using all training data (batch) for each iteration, stochastic gradient descent typically uses a mini-batch for each iteration of updating the parameters. It has sound theoretical foundations and works well in practice. It generally takes the following form

$$\theta^{(t+1)} = \theta^{(t)} - \eta(t)\nabla_\theta L(\theta^{(t)}) \tag{17}$$

  where θ is the parameter, η is the learning rate and L is the cost function.

- Momentum [RHW88]. Stochastic gradient descent usually has high variance and therefore slow convergence speed. In order to stabilize stochastic gradient descent, momentum method is used, which has the form

$$v^{(t+1)} = \alpha v^{(t)} - \eta(t)\nabla_\theta L(\theta^{(t)}) \tag{18}$$

$$\theta^{(t+1)} = \theta^{(t)} + v^{(t+1)} \tag{19}$$

  where $\alpha \in (0, 1)$ is a hyper-parameter and v is the momentum term.

- Initialization trick. It is well-known that The training of CNN is very sensitive to the initialization [LBOM12]. In [GB10], it is recommended to initialize the weight matrix of a neural network layer by drawing the values from a Gaussian distribution with zero mean and a specific variance,

$$\frac{2}{n_{in} + n_{out}}$$

  where n_{in} is the number of input neurons and n_{out} is the number of output neurons. In [SMG13], it is recommended to initialize the weight matrix of a neural network layer by an orthogonal or a semi-orthogonal matrix. The above two methods usually give better performance than a random standard normal initialization.

- Dropout. In [SHK+14], a technique called dropout was introduced. During training, it randomly shuts downs neurons with a probability p. At testing, the weights are multiplied by p. Dropout helps in reducing generalization error. However, it usually increases the training time.

- Batch normalization. In [IS15], a technique called batch normalization was introduced to accelerate the training of CNN. The method is shown in Algorithm 1. The motivation of batch normalization is to normalize the output of each neuron so as to stabilize the stochastic gradient descent algorithm. It is based on transformation of neurons so as to make a first-order optimization method close to a second-order one [RVL12].

Algorithm 1 Batch Normalization

1: **Input:** Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$ and parameters γ and β
2: **Output:** $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$
3: $\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i$
4: $\sigma^2_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2$
5: $\widehat{x}_i = \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}}$
6: $y_i = \gamma\widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i)$

Problems Despite that CNN is the state-of-the-art image classification method, it has several severe problems.

- Unrobustness. It has been shown that CNN could give complete different predictions of two perceptually similar images [SZS+13]. See Figure 17 for examples. It has also been shown that CNN is easy to fool [NYC14]. Given random noise or artificial images, CNN would predict them into some object classes with high confidence. See Figure 18 for examples.

- Slow training speed. CNN is known as notoriously difficult to train. In [KSH12], it has been shown that a CNN was trained for five to six days with two GPU cards on the ImageNet of 1.2 million images and 1000 classes.

- High dependence on labelled data. As discussed in the Section 2 CNN requires a large amount of labelled data, which is expensive and time-consuming to acquire. This problem is the focus of this thesis.

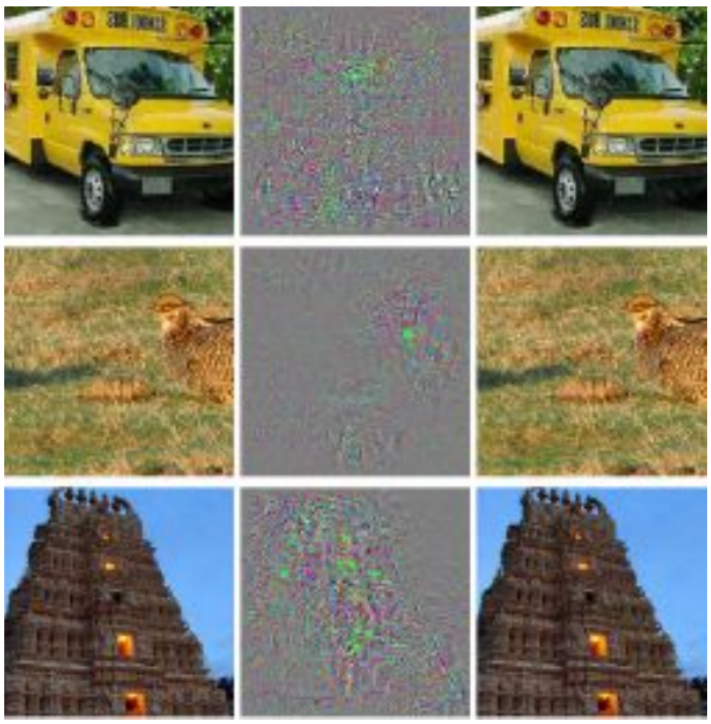


Figure 17: Adversary images. Left column: original images. Middle column: noise. Right column: Resulting images, to which CNN gives different predictions from the original images.[21]

2.2 Unsupervised Learning

There are two major branches of machine learning: supervised learning and unsupervised learning. They are divided according to the type of the training data and goals.

- Supervised learning. The training data is consisted of $\{(x, y)\}$, where x is the input and y is the target output. Define a cost function $c(y, \hat{y})$ where y is the true output and $\hat{y}$ is the estimated output. The goal of supervised learning is to estimate a function $f(x)$ for minimizing the cost $\int c(y, f(x))p(x, y)dxdy$. CNN belongs to this branch.

- Unsupervised learning. The training data is consisted of $\{x\}$, that is, only the inputs are observed and there is no target outputs. Unlike supervised learning, unsupervised learning does not have explicit task-specific goals. The role of

[21] Image from [SZS+13].

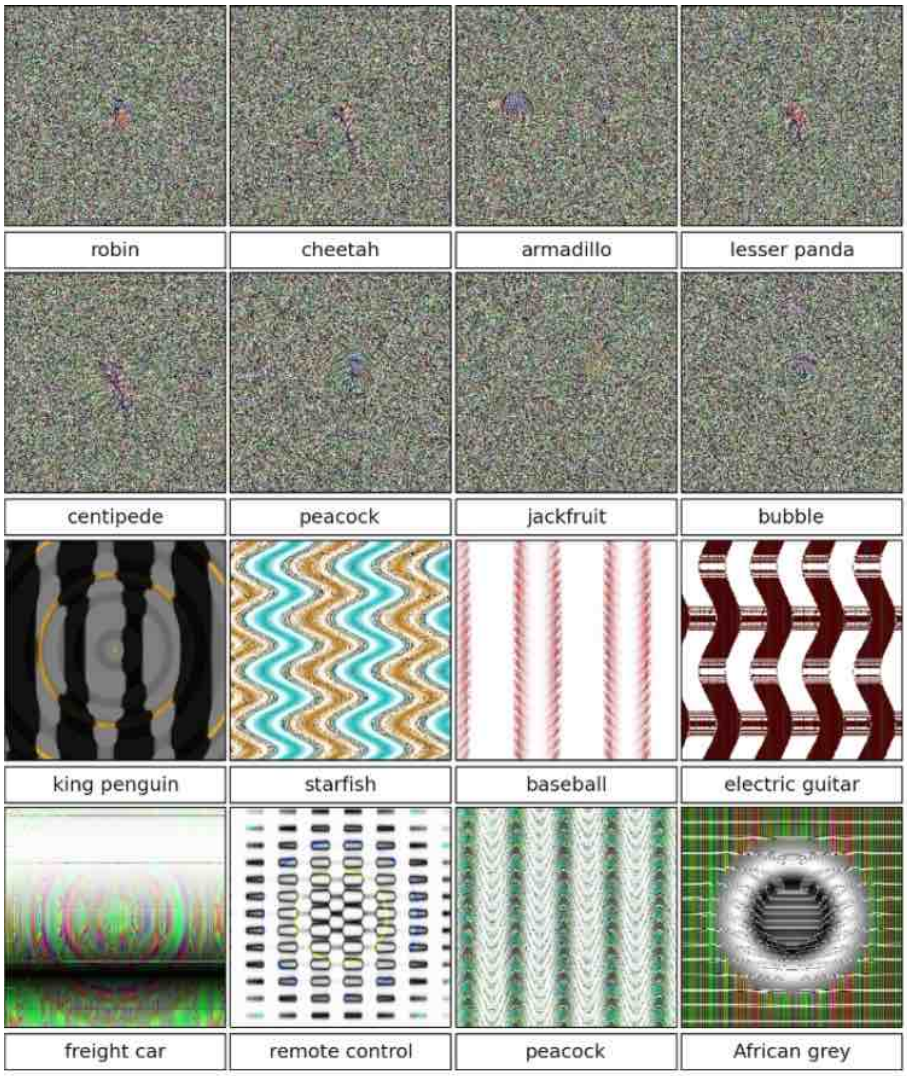


Figure 18: Images which fool CNN.[22]

unsupervised learning is rather indirect to the tasks at hand. We discuss the goals of unsupervised learning in details in below.

2.2.1 Goals of Unsupervised Learning

The goals of unsupervised include the following

- Structure discovering. Despite of variations of observed data, we believe there are hidden orders or laws underlying it (e.g. Kepler's laws of planetary motion). We call these orders or laws, structures. There are typically two kinds

[22]Image from [NYC14].

of structures: statistical and geometrical. Statistical structures are underlying rules which can be formulated in terms of probability and statistics concepts such as correlation, sparsity, independence and entropy. For example, using Bayesian networks structure to discover the casual dependency of variables. Geometric structures are underlying rules which can be formulated in terms of geometric or spatial concepts such as manifold, smoothness, locality and subspace.

- Visualization. The data we have at hand can be high-dimensional. However, we can only perceive two or three dimensional visual objects. In order to have a intuitive understanding of the data, we can use unsupervised learning algorithms such as PCA to reduce the dimensionality to two or three so as to visualize them.

- Denoising. There are corrupted or blurred images due to motion or noise in environment and image acqustion process. Unsupervised learning algorithms such as sparse code shrikage [Hyv99b] can be used to reduce these noise.

- Representation learning. A representation is a transformation of data. It is generally known that representation or features of data are critical to the subsequent tasks such as classification [Dom12]. For example, applying Independent Component Analysis (ICA) on natural image patches leads to Gabor-like features [HKO04], as will be shown in Section 2.2.3.

- Transfer learning. Transfer learning [PY10] is the use of the knowledge learned from one task to perform another task. Given enough amount of data, supervised learning is guaranteed to be sufficient for the task at and. But when the data is scarce, transfer learning is needed since it can "borrow" data or knowledge from one task to another.

In the following, we review several unsupervised learning algorithms, which are parts of our zero-shot learning method.

2.2.2 Principle Component Analysis (PCA)

Before we introduce Principle Component Analysis (PCA) [Pea01, Jol02], we first need to introduce Factor Analysis. We quote the definition of Factor Analysis from Wikipedia,

> *Factor Analysis is a statistical method used to describe variability among observed, correlated variables in terms of a potentially lower number of unobserved variables called factors.* -Wikipedia

In Factor Analysis, we assume the observed variables $\mathbf{x} = (x_1, ..., x_n)$ are generated by the following model

$$\mathbf{x} = \mathbf{A}\mathbf{s} + \mathbf{n} \tag{20}$$

where $\mathbf{s} = (s_1, ..., s_n)$ are the latent variables, $\mathbf{A}$ is the model parameter matrix and $\mathbf{n}$ are the noise variables. Here, $\mathbf{x}$ and $\mathbf{s}$ are assumed to have zero-mean. $\mathbf{s}$ are also assumed to be uncorrelated and have unit variance, in other words, white.

PCA is a special case of Factor Analysis. In PCA, $\mathbf{s}$ are assumed to be Gaussian and $\mathbf{n}$ are assumed to be zero (noise-free). There are several derivation of PCA.

- Maximizing variances. In this derivation, the objective function of the k-th principle component is

$$\max_{\mathbf{w}^{(k)}} \mathrm{Var}(\mathbf{x}^T\mathbf{w}^{(k)}) \tag{21}$$

$$\text{s.t.} \|\mathbf{w}^{(k)}\| = 1, \tag{22}$$

$$\mathbf{w}^{(j)T}\mathbf{w}^{(k)}, j = 1..., k-1. \tag{23}$$

- Minimizing reconstruction cost. In this derivation, the objective function of the k-th principle component is

$$\min_{\mathbf{w}^{(k)}} \sum_i \|\mathbf{x}_i - (\mathbf{w}^{(k)T}\mathbf{x}_i)\mathbf{w}^{(k)}\| \tag{24}$$

$$\text{s.t.} \|\mathbf{w}^{(k)}\| = 1, \tag{25}$$

$$\mathbf{w}^{(j)T}\mathbf{w}^{(k)}, j = 1..., k-1. \tag{26}$$

PCA is a classic method for dimensionality reduction or pre-processing. For a classification task, the input can be high dimensional and the intrinsic dimensionality of the data might be low. It is therefore necessary to reduce the dimensionality of the data for efficient computation and storage and sometimes reducing generalization error. For example, PCA can be employed for face recognition [TP+91]. The PCA components are therefore called Eigenface, as in Figure 19. Since we can reduce the dimensionality of data to two dimensional with PCA, PCA is commonly used for

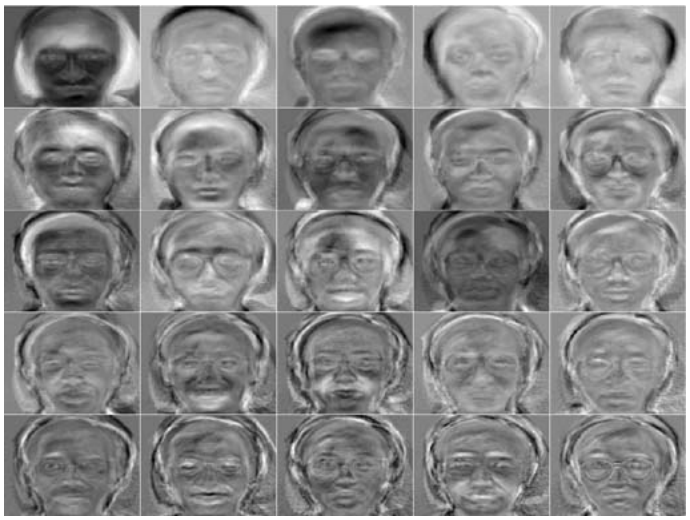


Figure 19: Eigenfaces.[23].

visualization of data. In the next section, we will show the visualization results of ImageNet object classes with PCA.

The solution of PCA can be obtained in several ways.

Eigendecomposition Let $\mathbf{C}$ denote the covariance matrix of $\mathbf{x}$, $\mathbf{E} = (\mathbf{e}_1, ..., \mathbf{e}_n)$ denote the matrix of eigenvectors of $\mathbf{C}$ and $\mathbf{D} = \text{diag}(\lambda_1, ..., \lambda_n)$ denote the diagonal matrix of eigenvalues of $\mathbf{C}$. The PCA matrix is $\mathbf{E}^T$, the whitening matrix is $\mathbf{U} = \mathbf{D}^{-1/2}\mathbf{E}^T$ and the whitened variables are $\mathbf{z} = \mathbf{U}\mathbf{x}$.

Singular Value Decomposition (SVD) Let $\mathbf{X}$ denote the data matrix. Using SVD, we have

$$\mathbf{X} = \mathbf{U}\mathbf{\Lambda}\mathbf{W} \tag{27}$$

where $\mathbf{U}$ and $\mathbf{W}$ are two orthogonal matrices and $\mathbf{\Lambda}$ is a diagonal matrix. $\mathbf{W}$ is the PCA matrix and the diagonal elements of $\mathbf{\Lambda}$ are the corresponding eigenvalues.

Online Algorithms There are several online PCA algorithms, which allow the PCA matrix to be learned with stream data. One of the most famous algorithm is Oja rule

$$\mathbf{w}(t+1) = \mathbf{w}(t) = \eta(t)y(t)(\mathbf{x}(t) - y(t)\mathbf{w}(t)) \tag{28}$$

[23]Image from `http://www.cs.princeton.edu/~cdecoro/eigenfaces/`.

where η is the learning rate, $\mathbf{x}$ is the input, $\mathbf{y} = \mathbf{w}^T\mathbf{x}$. However, it only computes the first principle component. For an extension of Oja rule to multiple components, see [OK85].

Another multiple principle components algorithm is the Generalized Hebbian Algorithm [San89]

$$\mathbf{W}(t+1) = \mathbf{W}(t) + \eta(t)(\mathbf{y}(t)\mathbf{x}(t)^T - \mathrm{LT}[\mathbf{y}(t)\mathbf{y}(t)^T]\mathbf{W}(t)) \tag{29}$$

where $\mathbf{W}$ is the PCA matrix, η is the learning rate, $\mathbf{x}$ is the input, $\mathbf{y} = \mathbf{W}\mathbf{x}$ and LT[] is the operation which sets all matrix elements above the diagonal equal to 0.

For another example, one can also use a QR-decomposition based algorithm

$$\mathbf{S}(t+1) = \mathbf{S}(t) + \eta(t)\mathbf{x}(t)\mathbf{x}^T(t)\mathbf{Q}(t) \tag{30}$$

$$\mathbf{S}(t+1) = \mathbf{Q}(t+1)\mathbf{R}(t+1) \quad \text{(QR decomposition)} \tag{31}$$

where $\mathbf{S}(0) = 0$ and $\mathbf{Q}$ is the PCA matrix.

2.2.3 Independent Component Analysis (ICA)

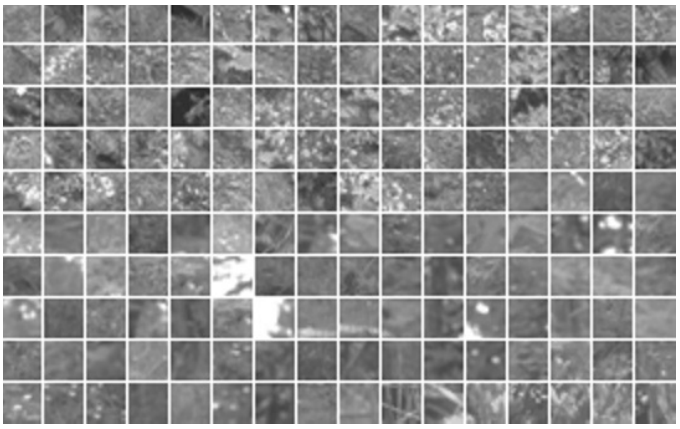
Independent Component Analysis (ICA) [JH91, Com94, HKO04] is another special case of Factor Analysis. In ICA, $\mathbf{s}$ are assumed to be non-Gaussian and independent and $\mathbf{n}$ are assumed to be zero. ICA seeks a demixing matrix $\mathbf{W}$ such that $\mathbf{s} = \mathbf{W}\mathbf{x}$ can be as independent as possible.

In the ICA model, there are two ambiguities which we cannot determine from the data

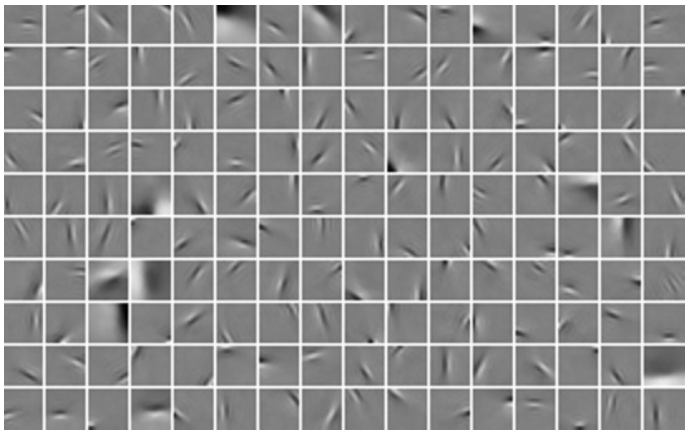
- We cannot determine the scaling of the independent components. ICA is ambiguous of scaling each component. If $\mathbf{W}$ is an ICA demixing matrix, then $\mathrm{diag}(\alpha_1, ..., \alpha_d)\mathbf{W}$ is also an ICA demixing matrix, where $\{\alpha_1, ..., \alpha_d\}$ are non-zero scaling constants of the components.
- We cannot determine the order of the independent components.

Despite of the ambiguities, ICA is still a useful model with many applications. We list some such applications in below.

- Feature extraction. ICA can be applied to natural image (patches) to learn the local features. These local feature will be useful for computer vision tasks



(a) Natural image patches



(b) ICA features

Figure 20: Features learned by ICA applied on natural image patches.[24]

such as classification. The ICA features resemble Gabor features. See Figure 20 for example.

- Blind signal separation. ICA can also be used for separating mixed signals. For example, in a cocktail party, many people's voices were recorded with several microphones. Since people speak simultaneously, the voices were mixed. With ICA, it is possible to recover the voice for each individual. See Figure 22 for illustration.

- Denoising. Another use of ICA is denoising. The idea is to apply ICA to the noisy signal. Then apply a thresholding nonlinear function to the transformed signals to remove the noise and transform the resulting signal back. The algorithm is called sparse coding shrinkage [Hyv99b].

[24]Image computed using ImageICA toolkit `http://research.ics.aalto.fi/ica/imageica/`.

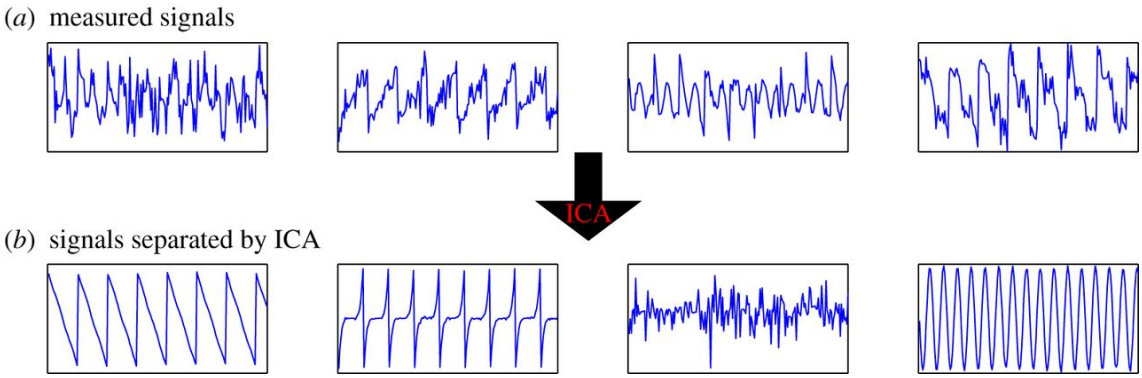


Figure 21: Blind slignal separation by ICA.[25]

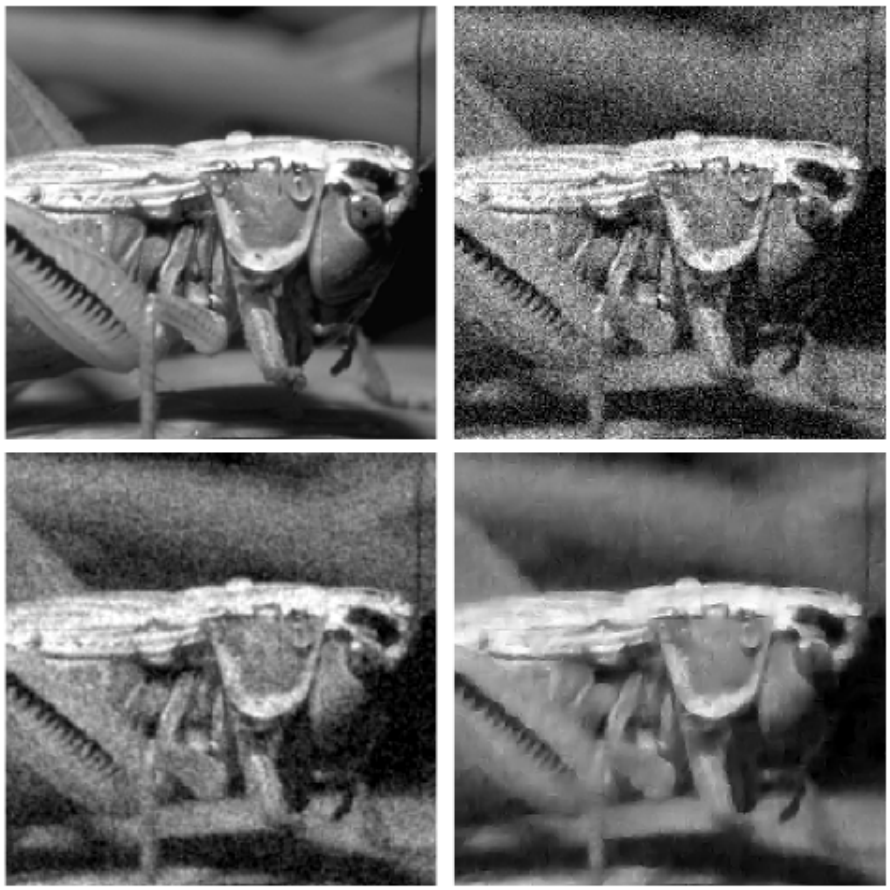


Figure 22: Image denoising by ICA. Top-left: The original image. Top-right: Noise added. Bottom-left: After Wiener filtering. Bottom-right: Results after ICA denoising.[27]

[25]Image from the lectures notes on unsupervised machine learning by Aapo Hyvarinen.
[27]Image from [Hyv99b].

The objective function of ICA can be formulated in several ways.

- Maximizing non-Gaussianity. We quote the definition of the central limit theorem from Wikipedia,

  > *given certain conditions, the arithmetic mean of a sufficiently large number of iterates of independent random variables, each with a well-defined expected value and well-defined variance, will be approximately normally distributed, regardless of the underlying distribution.* - Wikipedia.

  Roughly speaking, the more mixed/dependent the variables are, the more Gaussian they are. It implies a criterion for ICA, maximizing non-Gaussianity. Non-Gaussianity can be measure by kurtosis

  $$E(x^4)/E(x^2)^2 - 3 \tag{32}$$

  for variable x with zero mean, or negentropy.

  $$J(\mathbf{x}) = H(\mathbf{x}_{Gaussian}) - H(\mathbf{x}) \tag{33}$$

  where $\mathbf{x}_{Gaussian}$ is a Gaussian variable with the same correlation as variable $\mathbf{x}$ and $H()$ is the Shannon entropy defined as

  $$H(\mathbf{x}) = -\int p(\mathbf{x}) \log p(\mathbf{x}) d\mathbf{x} \tag{34}$$

  The exact analytic evaluation of negentropy is generally intractable. Therefore, approximations are needed. A classic approximation method is to use higher-order moments such that

  $$J(x) \approx \frac{1}{12} E(x^3)^2 + \frac{1}{48} \text{kurtosis}(x)^2 \tag{35}$$

  Another way is

  $$J(x) \propto (E(G(x)) - E(G(v)))^2 \tag{36}$$

  where $G(\cdot)$ is a nonlinear function and v is a Gaussian variable with zero mean and unit variance. The choice of $G(\cdot)$ depends on the distribution of x. Practically, one can choose

  $$G(x) = x^3 \tag{37}$$

$$G(x) = \log\cosh(x) \tag{38}$$

or

$$G(x) = -\exp(-x^2/2) \tag{39}$$

- Maximizing likelihood. We can also derive the likelihood function of ICA. For

$$\mathbf{x} = \mathbf{A}\mathbf{s} \tag{40}$$

we have

$$\begin{aligned} p_{\mathbf{x}}(\mathbf{x}) &= |\det(\mathbf{B})| p_{\mathbf{s}}(\mathbf{s}) && (41)\\ &= |\det(\mathbf{B})| \prod_i p_{s_i}(s_i) && (42)\\ &= |\det(\mathbf{B})| \prod_i p_{s_i}(\mathbf{b}_i^T\mathbf{x}) && (43) \end{aligned}$$

where $\mathbf{B} = \mathbf{A}^{-1}$. The likelihood function given data is

$$L(\mathbf{B}) = \prod_i \prod_{j=1}^{T} p_{s_i}(\mathbf{b}_i^T\mathbf{x}_j) |\det(\mathbf{B})| \tag{44}$$

And the log likelihood is

$$\log L(\mathbf{B}) = \sum_i \sum_{j=1}^{T} \log p_{s_i}(\mathbf{b}_i^T\mathbf{x}_j) + T\log|\det(\mathbf{B})| \tag{45}$$

Let

$$g_{s_i}(s_i) = \frac{\partial p_{s_i}(s_i)}{\partial s_i} \tag{46}$$

The gradient of the log likelihood function is

$$\frac{1}{T}\frac{\partial L(\mathbf{B})}{\partial \mathbf{B}} = [\mathbf{B}^T]^{-1} + E(g(\mathbf{B}\mathbf{x})\mathbf{x}^T) \tag{47}$$

- Maximizing mutual information. The mutual information for variables $\mathbf{y} = (y_1, ..., y_n)$ is defined as

$$\begin{aligned} I(\mathbf{y}) &= \sum_i H(y_i) - H(\mathbf{y}) && (48)\\ &= \sum_i H(\mathbf{b}_i\mathbf{x}) - H(\mathbf{x}) - \log|\det\mathbf{B}| && (49) \end{aligned}$$

We need to approximate the entropy here. Let

$$G_i(y_i) = \log p(y_i) \tag{50}$$

we have

$$I(\mathbf{y}) = -\sum_i E(G(y_i)) - \log|\det \mathbf{B}| - H(\mathbf{x}) \tag{51}$$

Since $H(\mathbf{x})$ does not depend on $\mathbf{B}$, maximizing mutual information is equivalent to maximizing likelihood.

FastICA A classic ICA algorithm is FastICA [Hyv99a], as in Algorithm 2. To obtain $\mathbf{W}$, we can first decompose it as

$$\mathbf{W} = \mathbf{VU}$$

where $\mathbf{U}$ is the whitening matrix and $\mathbf{V}$ is an orthogonal matrix, which can be learned by maximizing the non-Gaussianity or the likelihood function of $\mathbf{VUx}$. The non-Gaussianity can be measured by kurtosis or negentropy. If dimensionality reduction is required, we can take the d largest eigenvalues and the corresponding eigenvectors for the whitening matrix $\mathbf{U}$. As a result, the size of $\mathbf{U}$ is $d \times n$ and the size of $\mathbf{V}$ is $d \times d$.

Algorithm 2 FastICA

repeat
  for each column $\mathbf{v}$ of $\mathbf{V}$ **do**
    $\mathbf{v} \leftarrow E\{\mathbf{z}g(\mathbf{v}^T\mathbf{z})\} - E\{g'(\mathbf{v}^T\mathbf{z})\}\mathbf{v}$
  $\mathbf{V} \leftarrow \text{orthogonalize}(\mathbf{V})$
until $\mathbf{V}$ converges

Here $g(\cdot) = -\tanh(\cdot)$. $\mathbf{V}$ is initialized as a random orthogonal matrix. The assumption on the probability distribution of each s_i is a super-Gaussian distribution

$$\log p(s_i) = -\log\cosh(s_i) + \text{constant} \tag{52}$$

and therefore

$$g(s_i) = \frac{\partial}{\partial s_i}\log p(s_i) = -\tanh(s_i). \tag{53}$$

Another choice of $g(\cdot)$ is

$$g(s_i) = s_i \exp(-s_i^2/2). \tag{54}$$

There are several properties of FastICA.

- The convergence is cubic.
- No step size, in contrast to gradient based methods.
- Robustness.

SGD-based ICA Despite its fast convergence, FastICA is a batch algorithm which requires all the data to be loaded for computation in each iteration. Thus, it is unsuitable for large scale applications. To handle large scale datasets, we use a stochastic gradient descent (SGD) based ICA algorithm (described in the Appendix of [Hyv99a] and Section 3.4 of [Hyv99b]), as Algorithm 3.

Algorithm 3 SGD-based ICA

repeat
  for a random sample $\mathbf{z}(t)$ **do**
    $\mathbf{V} \leftarrow \mathbf{V} + \mu g(\mathbf{V}\mathbf{z}(t))\mathbf{z}(t)^T + \frac{1}{2}(\mathbf{I} - \mathbf{V}\mathbf{V}^T)\mathbf{V}^T$
until $\mathbf{V}$ converges

where μ is the learning rate, $g(\cdot)$ is a nonlinear function and $\mathbf{I}$ is an identity matrix.

Like FastICA, this SGD-based algorithm requires going through all data once to compute the whitening matrix $\mathbf{U}$. But unlike FastICA, this SGD-based algorithm does not require projection or orthogonalization in each step.

2.2.4 Multi-dimensional Scaling (MDS)

For n data with distance matrix $\mathbf{D}$, let D_{ij} denote the distance of data points i and j, the classic Multi-dimensional Scaling (MDS) [Tor58] embeds data points into an Euclidean space $\{\mathbf{x}_1, ..., \mathbf{x}_n\}$ such that the distortion of their distance in the Euclidean space is minimized. Formally,

$$\min_{\mathbf{x}_1,...,\mathbf{x}_n} \sum_{i<j} (\|\mathbf{x}_i - \mathbf{x}_j\| - D_{ij})^2 \tag{55}$$

The optimization problem is convex and can be solved by eigendecomposition. Classic MDS is used for embedding and visualizing similarity-based or distance-based data such as graphs.

The algorithm of classic MDS works as follows.

1. Let the centering matrix $\mathbf{J} = \mathbf{I} - n^{-1}\mathbf{I}\mathbf{I}^2$

2. Applying the double centering trick $\mathbf{B} = -\frac{1}{2}\mathbf{J}\mathbf{D}^2\mathbf{J}$

3. Extract k largest eigenvalues $\lambda_1, ..., \lambda_k$ and the corresponding eigenvectors $\mathbf{e}_1, ..., \mathbf{e}_k$.

4. Let $\mathbf{\Lambda}_k = \text{diag}(\lambda_1, ..., \lambda_k)$ and $\mathbf{E}_k = (\mathbf{e}_1, ..., \mathbf{e}_k)$. And embedding coordinates of the data points are $\mathbf{E}_k\mathbf{\Lambda}_k^{1/2}$.

2.2.5 Canonical Correlation Analysis (CCA)

Canonical Correlation Analysis [Hot36, HSST04] is a method of correlating linear relationships between two multidimensional variables. It has been used for content-based image retrieval. For two sets of vector data, $\{\mathbf{x}_1^{(1)}, ..., \mathbf{x}_n^{(1)}\}$ and $\{\mathbf{x}_1^{(2)}, ..., \mathbf{x}_n^{(2)}\}$, CCA projects them into a common space such that two sets have the maximal correlation or minimal distance. We present the two formulations of CCA in below.

- Maximizing correlation. For first pair of projections, we seek $(\mathbf{p}_1^{(1)}, \mathbf{p}_1^{(2)})$ that maximize

$$\text{corr}(\mathbf{p}_1^{(1)T}\mathbf{x}^{(1)}, \mathbf{p}_1^{(2)T}\mathbf{x}^{(2)}) \tag{56}$$

  For the second pair of projections, we seek $(\mathbf{p}_2^{(1)}, \mathbf{p}_2^{(2)})$ that are uncorrelated with the first pair of projections and maximizes the correlation of the projected variables. And so on.

- Minimizing distance.

$$\min_{\mathbf{P}^{(1)}, \mathbf{P}^{(2)}} \|\mathbf{P}^{(1)T}\mathbf{X}^{(1)} - \mathbf{P}^{(2)T}\mathbf{X}^{(2)}\|_F \tag{57}$$

$$\text{s.t.} \quad \mathbf{P}^{(k)T}\mathbf{C}_{kk}\mathbf{P}^{(k)} = \mathbf{I}, \quad \mathbf{p}_i^{(k)T}\mathbf{C}_{kl}\mathbf{p}_j^{(l)} = 0, \tag{58}$$

$$k, l = 1, 2, \quad k \neq l, \quad i, j = 1, ..., d, \tag{59}$$

  where $\mathbf{p}_i^{(k)}$ is the i-th column of $\mathbf{P}^{(k)}$ and $\mathbf{C}_{kl}$ is a covariance or cross-covariance matrix of $\{\mathbf{x}_1^{(1)}, ..., \mathbf{x}_n^{(1)}\}$ and/or $\{\mathbf{x}_1^{(2)}, ..., \mathbf{x}_n^{(2)}\}$.

The algorithm of CCA works as follows:

1. Let $\mathbf{E}_1 = \Sigma_{11}^{-1}\Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21}$ and $\mathbf{E}_2 = \Sigma_{22}^{-1}\Sigma_{21}\Sigma_{11}^{-1}\Sigma_{12}$.

2. Extract the top k eigenvectors of $\mathbf{E}_1$ and $\mathbf{E}_2$, respectively. The eigenvectors form the projection matrices $\mathbf{P}^{(1)}$ and $\mathbf{P}^{(2)}$.

2.3 Zero-shot Learning

Zero-shot learning [LEB08, LNH09] is a classification task in which some classes have no training data at all. We call the classes which have training data *seen classes* and those which have no training data *unseen classes*. One can use external knowledge of the classes, such as attributes, to build the relationship between the seen and the unseen classes. Then one can extrapolate the unseen classes by the seen classes.

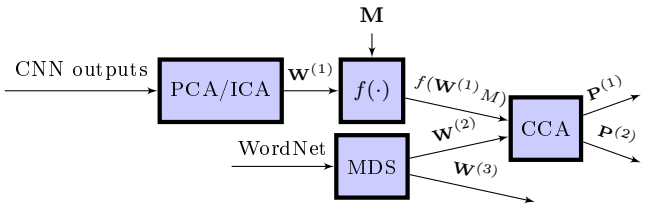
2.3.1 Previous Methods

Previous state-of-the-art large scale zero-shot learning methods are DeViSE [FCS+13] and conSE [NMB+14]. Both of them use the ImageNet of 1000 classes for training and the ImageNet of over 20000 classes for testing.

DeViSE In DeViSE, a CNN is first pre-trained on the ImageNet of 1000 classes. Then, 500-dimensional semantic features of both seen and unseen classes are obtained by running word2vec [MCCD13], an unsupervised word embedding algorithm, on Wikipedia. After that, the last (softmax) layer of the CNN is removed and the CNN is fine-tuned to predict the semantic features of the seen classes for each training image. In testing, when a new image arrives, the prediction is done by computing the cosine similarity of the CNN output vector and the semantic features of classes. In [FCS+13], it has also been shown that DeViSE could give more semantically reasonable errors for the seen classes.

ConSE In conSE, a CNN is first trained on the ImageNet of 1000 classes and 500-dimensional semantic features of the classes are obtained by running word2vec on Wikipedia, as in DeViSE. However, conSE does not require fine-tuning the CNN to predict the semantic features. The output vector in conSE is a convex combination of the semantic features, by the top activated neurons in the softmax layer. Its testing procedure is exactly the same as DeViSE. In [NMB+14], it was shown that conSE outperforms DeViSE in large scale zero-shot learning experiments.

Learning



Testing

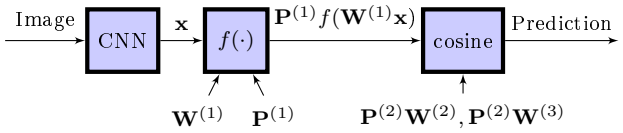


Figure 23: Our zero-shot learning method. $\mathbf{W}^{(1)}$ are the visual features of the seen classes. $\mathbf{W}^{(2)}$ are the semantic features of the seen classes. $\mathbf{W}^{(3)}$ are the semantic features of the unseen classes. $\mathbf{P}^{(1)}$ is the projection matrix from visual space to the common space. $\mathbf{P}^{(2)}$ is the projection matrix from semantic space to the common space. $f(\cdot)$ is the l_1 normalization. $\mathbf{M}$ are the mean vectors of the seen classes. $\mathbf{x}$ is the CNN output vector.

3 Proposed Zero-shot Learning Method

3.1 Overview

We proposed a new zero-shot learning method. Our method differs from DeViSE and conSE by using unsupervised learning algorithms to learn:

- Visual features of classes (with PCA and ICA).
- Semantic features of classes from the WordNet graph, instead of Wikipedia (with MDS).
- A common space between the visual and the semantic features (with CCA).

Our method works as follows. In the learning phase, first assume we have obtained the visual feature vectors $\mathbf{W}^{(1)} = (\mathbf{w}_1^{(1)}, ..., \mathbf{w}_n^{(1)})$ of n seen classes. Let $\mathbf{M} = (\mathbf{m}_1, ..., \mathbf{m}_n)$ denote the matrix of the mean outputs of a CNN of the seen classes. And $\mathbf{F} = f(\mathbf{W}^{(1)}\mathbf{M}) = (\mathbf{f}_1, ..., \mathbf{f}_n)$ are the transformed mean outputs of

the seen classes, where $f(\cdot)$ is a nonlinear function. Next, assume we have obtained the semantic feature vectors $\mathbf{W}^{(2)} = (\mathbf{w}_1^{(2)}, ..., \mathbf{w}_n^{(2)})$ of n seen classes and $\mathbf{W}^{(3)} = (\mathbf{w}_1^{(3)}, ..., \mathbf{w}_m^{(3)})$ of m unseen classes. Then we learn a bridge between the visual and the semantic representations of object classes via CCA to obtain two projection matrices $\mathbf{P}^{(1)}$ and $\mathbf{P}^{(2)}$.

In the testing phase, when a new image arrives, we first compute its CNN output $\mathbf{x}$. Then for $\mathbf{P}^{(1)T}(f(\mathbf{W}^{(1)}\mathbf{x}) - \frac{1}{n}\sum_i \mathbf{f}_i)$, we compute its k closest columns of $\mathbf{P}^{(2)T}\mathbf{W}^{(2)}$ (seen) and/or $\mathbf{P}^{(2)T}\mathbf{W}^{(3)}$ (unseen). The corresponding classes of these k columns are the top-k predictions. The closeness is measured by cosine similarity.

Each column of $\mathbf{W}^{(2)}$ and $\mathbf{W}^{(3)}$ is subtracted by $\frac{1}{n}\sum_i \mathbf{w}_i^{(2)}$. $\mathbf{M}$ is approximated by $\mathbf{I}$ and $f(\cdot)$ is the normalization of a vector or each column of a matrix by dividing its l_1 norm.

3.2 Learning Visual Features with PCA and ICA

In [HVD14], Hinton et al. showed that the softmax outputs of a trained neural network contain much richer information than just a one-hot classifier. Such a phenomenon is called *dark knowledge*. For input vector $\mathbf{y} = (y_1, ..., y_n)$, which is called logits in [HVD14], the softmax function produces output vector $\mathbf{x} = (x_1, ..., x_n)$ such that

$$x_i = \frac{\exp(y_i/T)}{\sum_j \exp(y_j/T)} \tag{60}$$

where T is the temperature parameter. The softmax function assigns positive probabilities to all classes since $x_i > 0$ for all i. Given a data point of a certain class as input, even when the probabilities of the incorrect classes are small, some of them are much larger than the others. For example, in a 4-class classification task (cow, dog, cat, car), given an image of a dog, while a hard target (class label) is $(0, 1, 0, 0)$, a trained neural network might output a soft target $(10^{-6}, 0.9, 0.1, 10^{-9})$. An image of a dog might have small chance to be misclassified as cat but it is much less likely to be misclassified as car. In [HVD14], a technique called knowledge distillation was introduced to further reveal the information in the softmax outputs. Knowledge distillation raises the temperature T in the softmax function to soften the outputs. For example, it transforms $(10^{-6}, 0.9, 0.1, 10^{-9})$ to $(0.015, 0.664, 0.319, 0.001)$ by raising temperature T from 1 to 3. It has been shown that adding the distilled soft targets in the objective function helps in reducing generalization error when training

a smaller model of an ensemble of models [HVD14]. Therefore, the outputs of a trained neural network are far from one-hot hard targets or random noise and they might contain rich statistical structures.

For the trained CNN model, we used GoogLeNet [SLJ+15], which is a deep neural network of 22 layers. It achieves the state-of-the-art results on ImageNet ILSVRC2014 challenge (6.67% top-5 error) and ILSVRC2012 validation set (32.9 % top-1 error and 12.1% top-5 error in our implementation). We used all the images in the ImageNet ILSVRC2012 training set to compute the ICA matrix using our SGD-based algorithm with mini-batch size 500. The learning rate was set to 0.005 and was halved every 10 epochs. The computation of CNN outputs was done with Caffe [JSD+14]. The ICA algorithm was ran with Theano [BBB+10].

To understand what is learned with PCA and ICA, we visualize the PCA and the ICA matrix. In the PCA matrix $\mathbf{E}^T$ or ICA matrix $\mathbf{W}$, each row corresponds to a PCA/ICA component and each column corresponds to an object class. The number of rows depends on the dimensionality reduction. The number of columns of $\mathbf{E}^T$ or $\mathbf{W}$ is 1000, corresponding to 1000 classes. After the ICA matrix was learned, each ICA component (a row of $\mathbf{W}$) was scaled to have unit l_2 norm. The scaling of each ICA component does not affect the ICA solution, as discussed in Section 2.2.3.

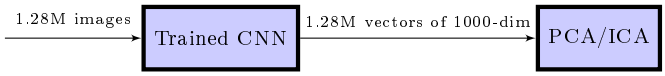


Figure 24: Learning visual features of object classes

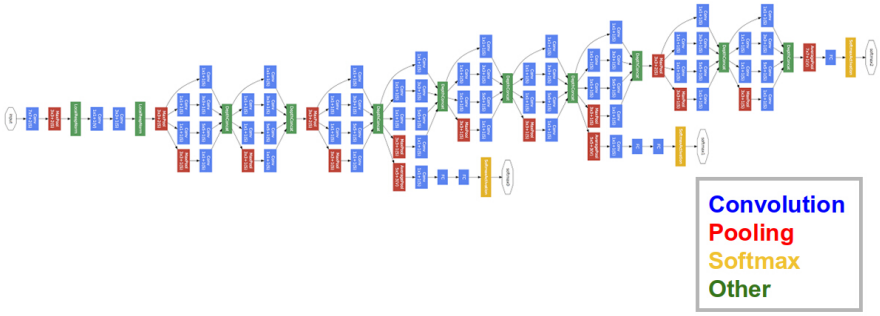


Figure 25: GoogLeNet.[28]

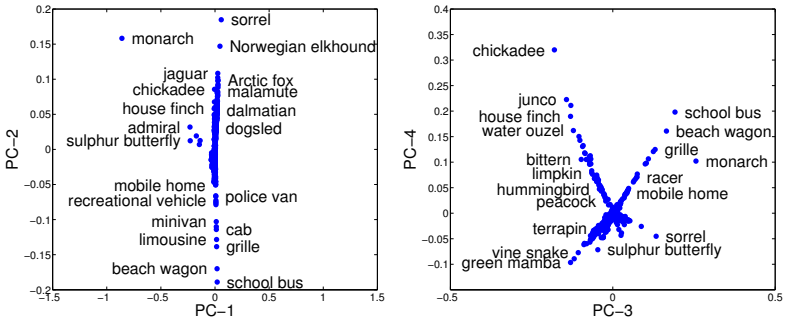
[28]Image from [SLJ+15].

Table 1: Object classes ranked by single components of PCA/ICA

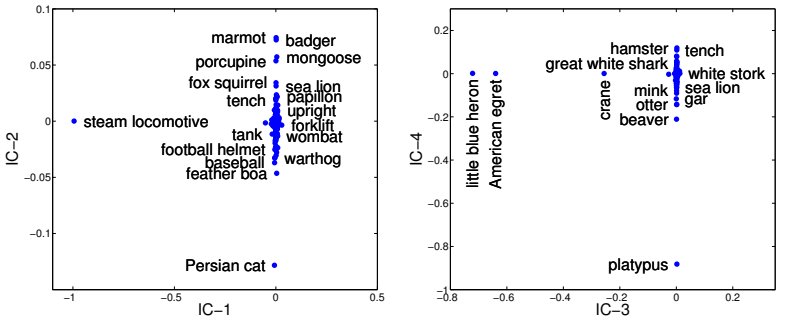
	1	2	3	4
PCA	mosque	killer whale	Model T	zebra
	shoji	beaver	strawberry	tiger
	trimaran	valley	hay	chickadee
	fire screen	otter	electric locomotive	school bus
	aircraft carrier	loggerhead	scoreboard	yellow lady's slipper
ICA	mosque	killer whale	Model T	zebra
	barn	grey whale	car wheel	tiger
	planetarium	dugong	tractor	triceratops
	dome	leatherback turtle	disk brake	prairie chicken
	palace	sea lion	barn	warthog

Table 2: Closest object classes in terms of visual and semantic similarity

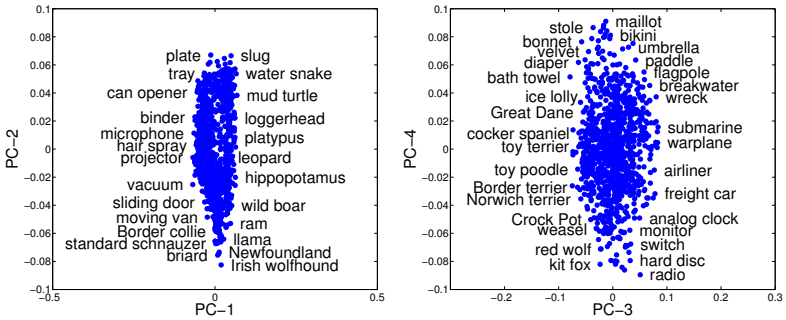
	Egyptian cat	soccer ball	mushroom	red wine
Visual	tabby cat	rugby ball	bolete	wine bottle
	tiger cat	croquet ball	agaric	beer glass
	tiger	racket	stinkhorn	goblet
	lynx	tennis ball	earthstar	measuring cup
	Siamese cat	football helmet	hen-of-the-woods	wine bottle
Semantic	Persian cat	croquet ball	cucumber	eggnog
	tiger cat	golf ball	artichoke	cup
	Siamese cat	baseball	cardoon	espresso
	tabby cat	ping-pong ball	broccoli	menu
	cougar	punching bag	cauliflower	meat loaf



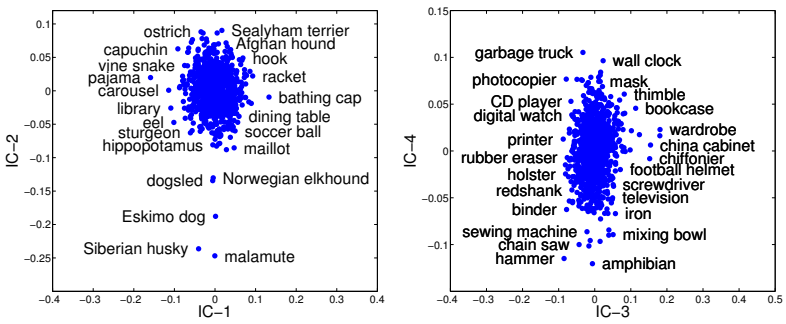
(a) PCA, softmax.



(b) ICA, softmax.



(c) PCA, normalized logits.



(d) ICA, normalized logits.

Figure 26: Label embedding of object class by PCA/ICA components. In each plot, each point is an object class and each axis is a PCA/ICA component (PC/IC). For visual clarity, only selected points are annotated with object class labels.

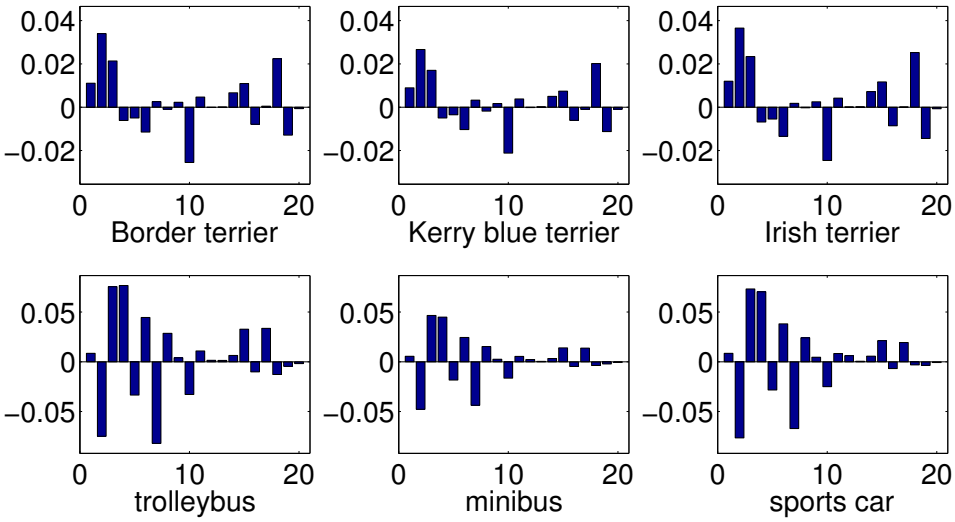
In Figure 26, we show the embedding of class labels by PCA and ICA. The horizontal and the vertical axes are two distinct rows of $\mathbf{E}^T$ or $\mathbf{W}$. Each point in the plot corresponds to an object class. There are 1000 points in each plot. Dimensionality is reduced from 1000 to 200 in ICA. In Figure 26 (a) and (b), we plot two pairs of the ICA/PCA components, learned with softmax outputs. In the PCA embedding, visually similar class labels are along some lines, but not the axes, while in the ICA embedding, they are along the axes. However, most points are clustered in the origin. In Figure 26 (c) and (d), we plot two pairs of the ICA/PCA components, learned with normalized logits outputs. We can see the class labels as points in the plots are more scattered.

In Figure 27, we show the PCA/ICA components of two sets of similar object classes: (1) *Border terrier*, *Lerry blue terrier*, and *Irish terrier*. (2) *trolleybus*, *minibus*, and *sports car*. Both PCA and ICA were learned on the softmax outputs and the dimensionality were reduced to 20 for better visualization. In Figure 27 (a), we see the PCA components of the object classes are distributed. While in Figure 27 (b), we see clearly some single components of ICA dominating. There are components representing "dog-ness" and "car-ness". Therefore, the ICA components are more interpretable.

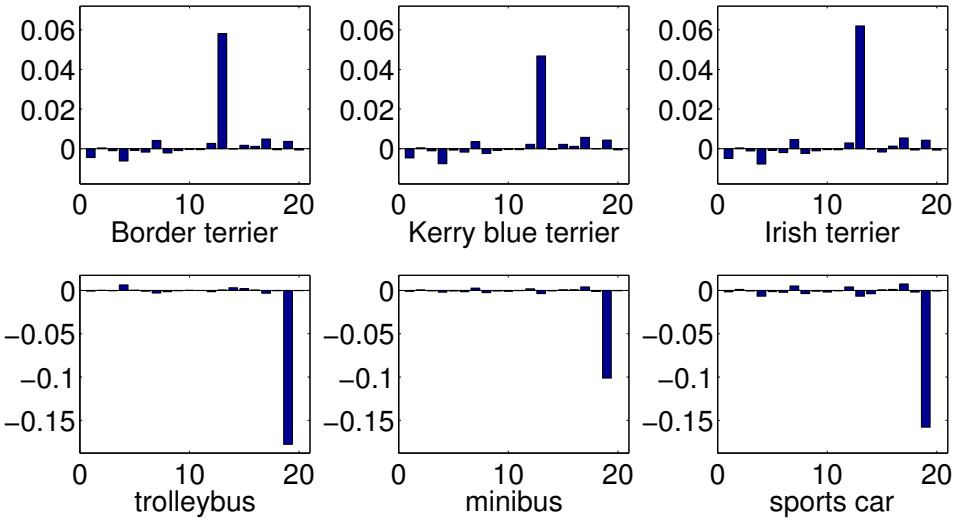
In Table 1, we show the top-5 object classes according to the value of PCA and ICA components. For the ease of comparison, we selected each PCA or ICA component which has the largest value for class *mosque*, *killer whale*, *Model T* or *zebra* among all components. We can see that the class labels ranked by ICA components are more visually similar than the ones by PCA components.

The PCA/ICA components can be interpreted as common features shared by visually similar object classes. From Figure 26 and Table 1, we can see the label embeddings of object classes by PCA/ICA components are meaningful since visually similar classes are close in the embeddings. Unlike [APHS13], these label embeddings can be unsupervisedly learned with a CNN trained with only one-hot class labels and without any hand annotated attribute label of the object classes, such as *has tail* or *lives in the sea*.

The visual-semantic similarity relationship was previously explored in [DF11], which shows some consistency between two similarities. Here we further explore it from another perspective. We define the visual and the semantic similarity in the following way. The visual similarity between two object classes is defined as cosine similarity of their PCA or ICA components (200-dim and learned with softmax), both of which



(a) PCA



(b) ICA

Figure 27: Bar plots of PCA/ICA components of object classes. Dimensionality was reduced to 20 for better visualization.

give the same results. The semantic similarity is defined based on the shortest path length [29] between two classes on the WordNet graph [Fel98]. In Table 2, we compare five closest classes of *Egyptian cat*, *soccer ball*, *mushroon* and *red wine* in terms of visual and semantic similarity. For *Egyptian cat*, both visual and semantic similarities give similar results. For *soccer ball*, the visual similarity gives *football helmet* which is quite distant in terms of semantic similarities. For *mushroom* and *red wine*, two similarities give very different closest object classes. The gap between two similarities is intriguing and therefore worth further exploration. In neuroscience literature, it is claimed that visual cortex representation favors visual rather than semantic similarity [BANP$^+$13].

3.3 Learning Semantic Features with MDS

For $\mathbf{W}^{(2)}$ and $\mathbf{W}^{(3)}$, we use the feature vectors by running classic Multi-dimensional Scaling (MDS), as introduced in Section 2.2.4, on a distance matrix of both seen and unseen classes. The distance between two classes is measured by one minus the similarity in the last subsection.

We use a new and simple method for obtaining the semantic features of classes, MDS on WordNet. Compare to Word2Vec on Wikipedia, this avoid three problems: (1) Word ambiguity. There are words which have multiple meanings and represent multiple classes. For example, there are two classes in ImageNet both named *fly*. One means the insect and the other means the fishermen's lure. (2) Multiple annotations. There are several classes which have multiple annotations. For example, class *lesser panda, red panda, panda, bear cat, cat bear, Ailurus fulgens*. With Word2Vec on Wikipedia, one would obtain multiple feature vectors for these classes. This is no principled way of selecting or combining them. (3) Heavy computation. Typically, Word2Vec on Wikipedia consumes a large amount of RAM and takes hours for computation. While classic MDS on the WordNet distance matrix of size 21842×21842[30] is much cheaper to compute. The computation of a 21632-dimensional MDS feature vector for each class was done in MATLAB with 8 Intel Xeon 2.5GHz cores within 12 minutes. A comparison of different semantic features of classes for zero-shot learning can be found in [ARW$^+$15]. However, MDS on WordNet was not compared in

[29] computed with the path_similarity() function in the NLTK tool `http://www.nltk.org/howto/wordnet.html`

[30] 21841 classes in ImageNet 2011fall plus class *teddy, teddy bear*. Class *teddy, teddy bear* (WordNet ID: n04399382) is in ImageNet ILSVRC2012 but not in ImageNet 2011fall.

[ARW+15] (only the raw WordNet distance matrix) and is a part of the contribution of this thesis.

3.4 Learning Visual-Semantic Common Space with CCA

Due to the visual-semantic similarity gap shown in the above, we learn a common space between the visual and the semantic representations of object classes via CCA, as introduced in Section 2.2.5, which seeks two projection matrices $\mathbf{P}^{(1)}$ and $\mathbf{P}^{(2)}$ such that

$$\min_{\mathbf{P}^{(1)},\mathbf{P}^{(2)}} \|\mathbf{P}^{(1)T}\mathbf{F} - \mathbf{P}^{(2)T}\mathbf{W}^{(2)}\|_F \tag{61}$$

$$\text{s.t.} \quad \mathbf{P}^{(k)T}\mathbf{C}_{kk}\mathbf{P}^{(k)} = \mathbf{I}, \quad \mathbf{p}_i^{(k)T}\mathbf{C}_{kl}\mathbf{p}_j^{(l)} = 0, \tag{62}$$

$$k, l = 1, 2, \quad k \neq l, \quad i, j = 1, ..., d, \tag{63}$$

where $\mathbf{p}_i^{(k)}$ is the i-th column of $\mathbf{P}^{(k)}$ and $\mathbf{C}_{kl}$ is a covariance or cross-covariance matrix of $\{\mathbf{f}_1, ..., \mathbf{f}_n\}$ and/or $\{\mathbf{w}_1^{(2)}, ..., \mathbf{w}_n^{(2)}\}$.

After projecting the visual features $\mathbf{F}$ and the semantic features $\mathbf{W}^{(2)}$ into a common space, the similarity comparison between images from the seen and the unseen becomes more sensible.

3.5 Experiments

Following the zero-shot learning experimental settings of DeViSE and conSE, we used a CNN trained on ImageNet ILSVRC2012 (1000 seen classes), and test our method to classify images in ImageNet 2011fall (20842 unseen classes [31], 21841 both seen and unseen classes). We use top-k accuracy (also called flat hit@k in [FCS+13, NMB+14]) measure, the percentage of test images in which a method's top-k predictions return the true label.

The CNN model we used is GoogLeNet. The sizes of the matrices in our methods: $\mathbf{W}^{(1)}$ is $k\times1000$, $\mathbf{W}^{(2)}$ is 21632×1000, $\mathbf{W}^{(3)}$ is 21632×20842, $\mathbf{P}^{(1)}$ is $k \times k$, $\mathbf{P}^{(2)}$ is $k\times21632$, $\mathbf{M}$ is 1000×1000 and $\mathbf{x}$ is 1000×1. We used $k = 100, 500, 900$ in our experiments. Although $\mathbf{W}^{(2)}$ and $\mathbf{W}^{(3)}$ are large matrices, we only need to compute once and store $\mathbf{P}^{(2)}\mathbf{W}^{(2)}$ and $\mathbf{P}^{(2)}\mathbf{W}^{(3)}$ of size $k\times1000$ and $k\times21632$, respectively.

[31] The class *teddy, teddy bear* is missing in ImageNet 2011fall, the correct number of classes is $21841-(1000-1) = 20842$ rather than 20841.

In Table 3, we show the results of different methods on ImageNet 2011fall. We compared random semi-orthogonal, PCA and ICA matrices as the visual features. Our method performs better when using PCA or ICA for the visual features than random features. And our method with random, PCA, or ICA features, achieves the state-of-the-art records on this zero-shot learning task.

In Table 4, we show the results of different methods on ImageNet ILSVRC2012 validation set of 1000 seen classes. While the goal here is not to classify images of seen classes, it is desirable to measure how much accuracy a zero-shot learning method would lose compared to the softmax baseline. Again, we can see that our method performs better using PCA or ICA for the visual features than random features.

In Table 5, we show the results of the three zero-shot learning methods on the test images selected in [NMB+14]. Same as conSE, our method gives correct or reasonable predictions. The correct labels are in blur color.

Table 3: Top-k accuracy in ImageNet 2011fall zero-shot learning task (%).

Test Set	#Classes	#Images	Method	Top-1	Top-2	Top-5	Top-10
Unseen	20842	12.9 million	DeViSE (500-dim)	0.8	1.4	2.5	3.9
			ConSE (500-dim)	1.4	2.2	3.9	5.8
			Our method (100-dim, random)	1.4	2.2	3.4	4.3
			Our method (100-dim, PCA)	1.6	2.7	4.6	6.4
			Our method (100-dim, ICA)	1.6	2.7	4.6	6.3
			Our method (500-dim, random)	**1.8**	2.9	5.0	6.9
			Our method (500-dim, PCA)	**1.8**	**3.0**	**5.2**	**7.3**
			Our method (500-dim, ICA)	**1.8**	**3.0**	**5.2**	**7.3**
			Our method (900-dim, random)	**1.8**	**3.0**	5.1	7.2
			Our method (900-dim, PCA)	**1.8**	**3.0**	**5.2**	**7.3**
			Our method (900-dim, ICA)	**1.8**	**3.0**	**5.2**	**7.3**
Both	21841	14.2 million	DeViSE (500-dim)	0.3	0.8	1.9	3.2
			ConSE (500-dim)	0.2	1.2	3.0	5.0
			Our method (100-dim, random)	**6.7**	8.2	10.0	11.1
			Our method (100-dim, PCA)	**6.7**	8.1	10.3	12.4
			Our method (100-dim, ICA)	**6.7**	8.1	10.4	12.4
			Our method (500-dim, random)	**6.7**	**8.5**	11.2	13.4
			Our method (500-dim, PCA)	**6.7**	**8.5**	**11.4**	**13.7**
			Our method (500-dim, ICA)	**6.7**	**8.5**	**11.4**	**13.7**
			Our method (900-dim, random)	**6.7**	**8.5**	**11.4**	**13.7**
			Our method (900-dim, PCA)	**6.7**	**8.5**	**11.4**	**13.7**
			Our method (900-dim, ICA)	**6.7**	**8.5**	**11.4**	**13.7**

[32]WordNet ID: n02077152. There are two classes named *fur seal* with different WordNet IDs.
[33]WordNet ID: n02077658.

Table 4: Top-k accuracy in ImageNet ILSVRC2012 validation set (%).

Test Set	#Classes	#Images	Method	Top-1	Top-2	Top-5	Top-10
Seen	1000	50000	Softmax baseline (1000-dim)	55.6	67.4	78.5	85.0
			DeViSE (500-dim)	53.2	65.2	76.7	83.3
			ConSE (500-dim)	54.3	61.9	68.0	71.6
			Our softmax baseline (1000-dim)	**67.1**	**78.8**	**87.9**	**92.2**
			Our method (100-dim, random)	67.0	74.6	77.8	79.1
			Our method (100-dim, PCA)	67.0	76.9	84.6	88.5
			Our method (100-dim, ICA)	67.0	76.9	84.6	88.5
			Our method (500-dim, random)	**67.1**	77.3	83.5	85.4
			Our method (500-dim, PCA)	**67.1**	78.2	86.2	89.4
			Our method (500-dim, ICA)	**67.1**	78.2	86.2	89.3
			Our method (900-dim, random)	**67.1**	78.3	86.0	88.6
			Our method (900-dim, PCA)	**67.1**	78.5	86.6	89.8
			Our method (900-dim, ICA)	**67.1**	78.4	86.5	89.8

Table 5: Predictions of test images of unseen classes (correct class labels are in blue)

Test Images	DeViSE	ConSE	Our Method
	water spaniel tea gown bridal gown spaniel tights	business suit **dress** hairpiece swimsuit kit	periwig mink tights quack-quack horsehair wig
	heron owl hawk bird of prey finch	**ratite** peafowl common spoonbill New World vulture Greek partridge	**ratite** kiwi moa elephant bird emu
	elephant turtle turtleneck flip-flop handcart	California sea lion **Steller sea lion** Australian sea lion South American sea lion eared seal	fur seal [32] eared seal fur seal [33] guadalupe fur seal Alaska fur seal
	**golden hamster** rhesus pipe shaker American mink	**golden hamster** rodent Eurasian hamster rhesus rabbit	**golden hamster** Eurasian hamster prairie dog skink mountain skink
	truck, motortruck skidder tank car automatic rifle trailer	flatcar truck tracked vehicle bulldozer wheeled vehicle	skidder bulldozer **farm machine** cultivator angledozer
	kernel littoral carillon Cabernet poodle	dog domestic cat schnauzer Belgian sheepdog domestic llama	mastiff Seeing Eye dog guide dog **alpaca** domestic llama

3.6 Discussion

The results of our experiments show that in our method the PCA or ICA matrix as visual features of object classes performs better than a random matrix. Therefore, these visual features, learned by PCA and ICA on the outputs of CNN, are indeed effective for the subsequent tasks. The results also show that PCA and ICA give the essentially same classification accuracy. Therefore, in practice we can use PCA instead of ICA, which has much higher computational costs. For a more comprehensive discussion on PCA vs. ICA for recognition tasks, see [AVHH07].

4 Conclusions

The outputs of a neural network contains rich information. It has been claimed that one can determine a neural network architecture by observing its outputs given arbitrary inputs [FM94]. Also, it has been shown that one can reconstruct the whole image to some degree with only its CNN outputs [DB15]. And smooth regularization on the output distribution of a neural network can help in reducing generalization error in both supervised and semi-supervised settings [MMK+15].

CNN achieves the state-of-the-art results on many computer vision tasks such as image classification and object detection. However, despite many efforts of visualizing and understanding CNN [ZF14, SVZ14, ZKL+15], it still reminds a black-box method. In this thesis, we attempted to understand CNN by unsupervised learning. CNN was trained with only one-hot targets, which means we assumed object classes are equally similar. We never told CNN which classes more similar. But unsupervised learning on CNN outputs reveals the visual similarity of object classes. We hope this finding can shed some lights on the object representation in CNN.

We also showed that there is a gap between the visual similarity of object classes in CNN and the semantic similarity of object classes in our knowledge graph. Therefore, a bridge should be built, in order to achieve consistent mapping between visual and semantic representations.

Supervised learning alone cannot deal with unseen classes since there is no training data. By using external knowledge and unsupervised learning algorithms, we can leverage supervised learning so as to make reasonable predictions on the unseen classes while maintaining the compatibility with the seen classes, that is, zero-shot learning. In this thesis, we proposed a new zero-shot learning method, which achieves

the state-of-the-art results on the ImageNet of over 20000 classes.

Finally, we list several open questions for the future research.

- How to nonlinearly extract the visual and the semantic features of the object classes? In our proposed zero-shot learning method, we only used linear algorithm: PCA and ICA for the visual features and MDS for the semantic features. There are various nonlinear algorithms such as kernel PCA [SSM98], kernel ICA [BJ03] and IsoMap [TDSL00]. However, the issues with nonlinear models are scalability and the choice of nonlinearity.

- How to nonlinearly build the bridge between the visual and the semantic features of the object classes? We used linear CCA for projecting the visual and the semantic features into a common space. There are nonlinear CCA algorithms based on kernel method [HSST04] and neural networks [Hsi00, AABL13]. Again, the issues are scalability and the choice of nonlinearity.

- How to incorporate zero-shot learning with additional labelled data? In practice, we often could obtain some additional labelled data for the unseen classes. A naive way of incorporating the labelled data is to re-train the CNN model and apply a zero-shot learning method on the remaining unseen classes. However, the re-training is rather time and resource consuming. We look forward to seeing a more efficient and principled way to integrate additional labelled data with an existing zero-shot learning method.

References

AABL13 Andrew, G., Arora, R., Bilmes, J. and Livescu, K., Deep canonical correlation analysis. *Proceedings of the 30th International Conference on Machine Learning*, 2013, pages 1247–1255.

APHS13 Akata, Z., Perronnin, F., Harchaoui, Z. and Schmid, C., Label-embedding for attribute-based classification. *IEEE Conference on Computer Vision and Pattern Recognition*.

ARW+15 Akata, Z., Reed, S., Walter, D., Lee, H. and Schiele, B., Evaluation of output embeddings for fine-grained image classification. *IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

AT13 Andreopoulos, A. and Tsotsos, J. K., 50 years of object recognition: Directions forward. *Computer Vision and Image Understanding*, 117,8(2013), pages 827–891.

AVHH07 Asuncion Vicente, M., Hoyer, P. O. and Hyvärinen, A., Equivalence of some common linear feature extraction techniques for appearance-based object recognition tasks. *PAMI*.

BANP+13 Baldassi, C., Alemi-Neissi, A., Pagan, M., DiCarlo, J. J., Zecchina, R. and Zoccolan, D., Shape similarity, better than semantic membership, accounts for the structure of visual object representations in a population of monkey inferotemporal neurons. *PLoS computational biology*.

BBB+10 Bergstra, J., Breuleux, O., Bastien, F., Lamblin, P., Pascanu, R., Desjardins, G., Turian, J., Warde-Farley, D. and Bengio, Y., Theano: a CPU and GPU math expression compiler. *Proceedings of the Python for Scientific Computing Conference*.

BCV13 Bengio, Y., Courville, A. and Vincent, P., Representation learning: A review and new perspectives. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35,8(2013), pages 1798–1828.

BJ03 Bach, F. R. and Jordan, M. I., Kernel independent component analysis. *The Journal of Machine Learning Research*, 3, pages 1–48.

BPL10 Boureau, Y.-L., Ponce, J. and LeCun, Y., A theoretical analysis of feature pooling in visual recognition. *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, 2010, pages 111–118.

CHY+14 Cadieu, C. F., Hong, H., Yamins, D. L., Pinto, N., Ardila, D., Solomon, E. A., Majaj, N. J. and DiCarlo, J. J., Deep neural networks rival the representation of primate it cortex for core visual object recognition. *PLoS computational biology*.

CMS12 Ciresan, D., Meier, U. and Schmidhuber, J., Multi-column deep neural networks for image classification. *IEEE Conference on Computer Vision and Pattern Recognition*.

Com94 Comon, P., Independent component analysis, a new concept? *Signal processing*, 36,3(1994), pages 287–314.

CV95 Cortes, C. and Vapnik, V., Support-vector networks. *Machine learning*, 20,3(1995), pages 273–297.

DB15 Dosovitskiy, A. and Brox, T., Inverting convolutional networks with convolutional networks. *arXiv:1506.02753*.

DDS+09 Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K. and Fei-Fei, L., Imagenet: A large-scale hierarchical image database. *IEEE Conference on Computer Vision and Pattern Recognition*.

DF11 Deselaers, T. and Ferrari, V., Visual and semantic similarity in imagenet. *IEEE Conference on Computer Vision and Pattern Recognition*.

Dom12 Domingos, P., A few useful things to know about machine learning. *Communications of the ACM*, 55,10(2012), pages 78–87.

FCS+13 Frome, A., Corrado, G. S., Shlens, J., Bengio, S., Dean, J., Mikolov, T. et al., Devise: A deep visual-semantic embedding model. *Advances in Neural Information Processing Systems*.

Fel98 Fellbaum, C., *WordNet: An Electronic Lexical Database*. Bradford Books, 1998.

FFP05 Fei-Fei, L. and Perona, P., A bayesian hierarchical model for learning natural scene categories. *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 2. IEEE, 2005, pages 524–531.

FM94 Fefferman, C. and Markel, S., Recovering a feed-forward net from its output. *NIPS*, 1994.

Fuk88 Fukushima, K., Neocognitron: A hierarchical neural network capable of visual pattern recognition. *Neural networks*, 1,2(1988), pages 119–130.

GB10 Glorot, X. and Bengio, Y., Understanding the difficulty of training deep feedforward neural networks. *International conference on artificial intelligence and statistics*, 2010, pages 249–256.

GDDM14 Girshick, R., Donahue, J., Darrell, T. and Malik, J., Rich feature hierarchies for accurate object detection and semantic segmentation. *IEEE Conference on Computer Vision and Pattern Recognition*.

HH00 Hyvärinen, A. and Hoyer, P., Emergence of phase-and shift-invariant features by decomposition of natural images into independent feature subspaces. *Neural computation*, 12,7(2000), pages 1705–1720.

HHI01 Hyvärinen, A., Hoyer, P. and Inki, M., Topographic independent component analysis. *Neural computation*, 13,7(2001), pages 1527–1558.

HKO04 Hyvärinen, A., Karhunen, J. and Oja, E., *Independent component analysis*. John Wiley & Sons, 2004.

Hot36 Hotelling, H., Relations between two sets of variates. *Biometrika*.

HOT06 Hinton, G. E., Osindero, S. and Teh, Y.-W., A fast learning algorithm for deep belief nets. *Neural computation*, 18,7(2006), pages 1527–1554.

Hsi00 Hsieh, W. W., Nonlinear canonical correlation analysis by neural networks. *Neural Networks*, 13,10(2000), pages 1095–1105.

HSST04 Hardoon, D., Szedmak, S. and Shawe-Taylor, J., Canonical correlation analysis: An overview with application to learning methods. *Neural computation*.

Hub82 Hubel, D. H., Exploration of the primary visual cortex, 1955–78. *Nature*, 299,5883(1982), pages 515–524.

HVD14 Hinton, G. E., Vinyals, O. and Dean, J., Distilling the knowledge in a neural network. *NIPS Deep Learning Workshop*.

Hyv99a Hyvarinen, A., Fast and robust fixed-point algorithms for independent component analysis. *IEEE Transactions on Neural Networks*.

Hyv99b Hyvarinen, A., Sparse code shrinkage: Denoising of nongaussian data by maximum likelihood estimation. *Neural computation*.

IS15 Ioffe, S. and Szegedy, C., Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*.

JH91 Jutten, C. and Herault, J., Blind separation of sources, part i: An adaptive algorithm based on neuromimetic architecture. *Signal processing*, 24,1(1991), pages 1–10.

Jol02 Jolliffe, I., *Principal component analysis*. Wiley Online Library, 2002.

JSD+14 Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S. and Darrell, T., Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*.

KH09 Krizhevsky, A. and Hinton, G., Learning multiple layers of features from tiny images, 2009.

KSH12 Krizhevsky, A., Sutskever, I. and Hinton, G. E., Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*.

LBBH98 LeCun, Y., Bottou, L., Bengio, Y. and Haffner, P., Gradient-based learning applied to document recognition. *Proceedings of the IEEE*.

LBH15 LeCun, Y., Bengio, Y. and Hinton, G., Deep learning. *Nature*.

LBOM12 LeCun, Y. A., Bottou, L., Orr, G. B. and Müller, K.-R., Efficient backprop. In *Neural networks: Tricks of the trade*, Springer, 2012, pages 9–48.

LEB08 Larochelle, H., Erhan, D. and Bengio, Y., Zero-data learning of new tasks. *AAAI Conference on Artificial Intelligence*.

LLZ+11 Lin, Y., Lv, F., Zhu, S., Yang, M., Cour, T., Yu, K., Cao, L. and Huang, T., Large-scale image classification: fast feature extraction and svm training. *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*. IEEE, 2011, pages 1689–1696.

LNH09 Lampert, C. H., Nickisch, H. and Harmeling, S., Learning to detect unseen object classes by between-class attribute transfer. *IEEE Conference on Computer Vision and Pattern Recognition*.

Low99 Lowe, D. G., Object recognition from local scale-invariant features. *IEEE international conference on Computer vision*, 1999.

LSD15 Long, J., Shelhamer, E. and Darrell, T., Fully convolutional networks for semantic segmentation. *IEEE Conference on Computer Vision and Pattern Recognition*.

LSP06 Lazebnik, S., Schmid, C. and Ponce, J., Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 2. IEEE, 2006, pages 2169–2178.

MCCD13 Mikolov, T., Chen, K., Corrado, G. and Dean, J., Efficient estimation of word representations in vector space. *Internation Conference on Learning Representation.*

MHL13 Mathieu, M., Henaff, M. and LeCun, Y., Fast training of convolutional networks through ffts. *arXiv preprint arXiv:1312.5851.*

MLY14 Min Lin, Q. C. and Yan, S., Network in network. *Internation Conference on Learning Representation.*

MMK+15 Miyato, T., Maeda, S.-i., Koyama, M., Nakae, K. and Ishii, S., Distributional smoothing with virtual adversarial training. *arXiv:1507.00677*, 1050.

NMB+14 Norouzi, M., Mikolov, T., Bengio, S., Singer, Y., Shlens, J., Frome, A., Corrado, G. S. and Dean, J., Zero-shot learning by convex combination of semantic embeddings. *International Conference on Learning Representation.*

NYC14 Nguyen, A., Yosinski, J. and Clune, J., Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. *arXiv preprint arXiv:1412.1897.*

OF97 Olshausen, B. A. and Field, D. J., Sparse coding with an overcomplete basis set: A strategy employed by v1? *Vision research*, 37,23(1997), pages 3311–3325.

OK85 Oja, E. and Karhunen, J., On stochastic approximation of the eigenvectors and eigenvalues of the expectation of a random matrix. *Journal of mathematical analysis and applications*, 106,1(1985), pages 69–84.

Pap66 Papert, S., The summer vision project.

PD07 Perronnin, F. and Dance, C., Fisher kernels on visual vocabularies for image categorization. *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on.* IEEE, 2007, pages 1–8.

Pea01 Pearson, K., On lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2,11(1901), pages 559–572.

Per09 Perona, P., Visual recognition circa 2008. *Object categorization: computer and human vision perspectives. Cambridge University Press, Cambridge/New York*, pages 55–68.

PPHM09 Palatucci, M., Pomerleau, D., Hinton, G. E. and Mitchell, T. M., Zero-shot learning with semantic output codes. *Advances in Neural Information Processing Systems*.

PSM10 Perronnin, F., Sánchez, J. and Mensink, T., Improving the fisher kernel for large-scale image classification. In *Computer Vision–ECCV 2010*, Springer, 2010, pages 143–156.

PY10 Pan, S. J. and Yang, Q., A survey on transfer learning. *Knowledge and Data Engineering, IEEE Transactions on*, 22,10(2010), pages 1345–1359.

RHW88 Rumelhart, D. E., Hinton, G. E. and Williams, R. J., Learning representations by back-propagating errors. *Cognitive modeling*, 5, page 3.

RM51 Robbins, H. and Monro, S., A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407.

RP99 Riesenhuber, M. and Poggio, T., Hierarchical models of object recognition in cortex. *Nature neuroscience*, 2,11(1999), pages 1019–1025.

RSS11 Rohrbach, M., Stark, M. and Schiele, B., Evaluating knowledge transfer and zero-shot learning in a large-scale setting. *CVPR*, 2011.

RVL12 Raiko, T., Valpola, H. and LeCun, Y., Deep learning made easier by linear transformations in perceptrons. *International Conference on Artificial Intelligence and Statistics*, 2012, pages 924–932.

San89 Sanger, T. D., Optimal unsupervised learning in a single-layer linear feedforward neural network. *Neural networks*, 2,6(1989), pages 459–473.

SGMN13 Socher, R., Ganjoo, M., Manning, C. D. and Ng, A., Zero-shot learning through cross-modal transfer. *NIPS*, 2013.

SHK+14 Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. and Salakhutdinov, R., Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15,1(2014), pages 1929–1958.

SLJ+15 Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V. and Rabinovich, A., Going deeper with convolutions. *IEEE Conference on Computer Vision and Pattern Recognition*.

SMG13 Saxe, A. M., McClelland, J. L. and Ganguli, S., Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv preprint arXiv:1312.6120*.

SP11 Sánchez, J. and Perronnin, F., High-dimensional signature compression for large-scale image classification. *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*. IEEE, 2011, pages 1665–1672.

SR15 Shaoqing Ren, Kaiming He, R. G. J. S., Faster R-CNN: Towards real-time object detection with region proposal networks. *Advances in Neural Information Processing*.

SSM98 Schölkopf, B., Smola, A. and Müller, K.-R., Nonlinear component analysis as a kernel eigenvalue problem. *Neural computation*, 10,5(1998), pages 1299–1319.

SSSSC11 Shalev-Shwartz, S., Singer, Y., Srebro, N. and Cotter, A., Pegasos: Primal estimated sub-gradient solver for svm. *Mathematical programming*, 127,1(2011), pages 3–30.

SVZ14 Simonyan, K., Vedaldi, A. and Zisserman, A., Deep inside convolutional networks: Visualising image classification models and saliency maps. *International Conference on Learning Representations*.

SZS+13 Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I. and Fergus, R., Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*.

TDSL00 Tenenbaum, J. B., De Silva, V. and Langford, J. C., A global geometric framework for nonlinear dimensionality reduction. *Science*, 290,5500(2000), pages 2319–2323.

TMJ+10 Turaga, S. C., Murray, J. F., Jain, V., Roth, F., Helmstaedter, M., Briggman, K., Denk, W. and Seung, H. S., Convolutional networks can learn to generate affinity graphs for image segmentation. *Neural Computation*.

Tor58 Torgerson, W. S., Theory and methods of scaling.

TP+91 Turk, M., Pentland, A. P. et al., Face recognition using eigenfaces. *Computer Vision and Pattern Recognition, 1991. Proceedings CVPR'91., IEEE Computer Society Conference on*. IEEE, 1991, pages 586–591.

VEAF92 Van Essen, D. C., Anderson, C. H. and Felleman, D. J., Information processing in the primate visual system: an integrated systems perspective. *Science*, 255,5043(1992), pages 419–423.

Wer82 Werbos, P. J., Applications of advances in nonlinear sensitivity analysis. In *System modeling and optimization*, Springer, 1982, pages 762–770.

XWCL15 Xu, B., Wang, N., Chen, T. and Li, M., Empirical evaluation of rectified activations in convolutional network. *arXiv preprint arXiv:1505.00853*.

ZF14 Zeiler, M. D. and Fergus, R., Visualizing and understanding convolutional networks. *European Conference on Computer Vision*.

ZKL+15 Zhou, B., Khosla, A., Lapedriza, A., Oliva, A. and Torralba, A., Object detectors emerge in deep scene cnns. *International Conference on Learning Representation*.